

# AN EFFICIENT TREE DECOMPOSITION METHOD FOR PERMANENTS AND MIXED DISCRIMINANTS

DIEGO CIFUENTES AND PABLO A. PARRILO

**ABSTRACT.** We present an efficient algorithm to compute permanents, mixed discriminants and hyperdeterminants of structured matrices and multidimensional arrays (tensors). We describe the sparsity structure of an array in terms of a graph, and we assume that its treewidth, denoted as  $\omega$ , is small. Our algorithm requires  $\tilde{O}(n 2^\omega)$  arithmetic operations to compute permanents, and  $\tilde{O}(n^2 + n 3^\omega)$  for mixed discriminants and hyperdeterminants. We finally show that mixed volume computation continues to be hard under bounded treewidth assumptions.

## 1. INTRODUCTION

The *permanent* of a  $n \times n$  matrix  $M$  is defined as

$$\text{Perm}(M) := \sum_{\pi} \prod_{i=1}^n M_{i,\pi(i)}$$

where the sum is over all permutations  $\pi$  of the numbers  $1, \dots, n$ . Computing the permanent is #P-hard [35], which means that it is unlikely that it can be done efficiently for arbitrary matrices. As a consequence, research on this problem tends to fall into two categories: algorithms to approximate the permanent, and exact algorithms that assume some structure of the matrix. This paper lies in the second category. We further study related problems in structured higher dimensional arrays, such as mixed discriminants, hyperdeterminants and mixed volumes.

The sparsity pattern of a matrix  $M$  can be seen as the bipartite adjacency matrix of some bipartite graph  $G$ . This bipartite graph fully encodes the structure of the matrix. We assume here that the treewidth  $\omega$  of  $G$  is small. Even though it is hard to determine the treewidth of a graph, there are many good heuristics and approximations that justify our assumption [12]. We show an algorithm to compute  $\text{Perm}(M)$  in  $\tilde{O}(n 2^\omega)$  arithmetic operations. In this paper, the notation  $\tilde{O}$  ignores polynomial factors in  $\omega$ . We note that the algorithm can be used over any commutative ring.

The permanent of a matrix can be generalized in several ways. In particular, given a list of  $n$  matrices of size  $n \times n$ , its *mixed discriminant* generalizes both the permanent and the determinant [4, 21]. Our algorithm for the permanent extends in a natural way to compute the mixed discriminant. The natural structure to represent the sparsity pattern in this case is a tripartite (i.e., 3-colorable) graph. The running time of the

---

*Date:* July 14, 2015.

*Key words and phrases.* Permanent, Structured array, Mixed discriminant, Treewidth.

resulting algorithm is  $\tilde{O}(n^2 + n 3^\omega)$ , where  $\omega$  is the treewidth of such graph. In particular, this algorithm can compute the determinant of a matrix in the same time.

More generally, our methods extend to generalized determinants/permanents on tensors. A special case of interest is the *multidimensional permanent* [2, 18, 34]. Another interesting case is the first Cayley *hyperdeterminant*, also known as Pascal determinant, which is the simplest generalization of the determinant to higher dimensions [5, 14, 25]. Note that unlike the determinant, the hyperdeterminant is  $\#P$ -hard, in particular because it contains mixed discriminants as a special case [21, 22].

Given a set of  $n$  polytopes in  $\mathbb{R}^n$ , its mixed volume provides a geometric generalization of the permanent and the determinant [30]. We focus on the special case of mixed volume of  $n$  zonotopes. Although there is no “natural” graph to represent the structure of a set of zonotopes, we associate to it a bipartite graph that, when the mixed volume restricts to a permanent, corresponds to the matrix graph described above. This allows us to give a simple application for mixed volumes of zonotopes with few nonparallel edges. Nevertheless, we show that mixed volumes remain hard to compute in the general case, even if this bipartite graph has treewidth 1 and the zonotopes have only 3 nonzero coordinates.

The diagram of Figure 1 summarizes the scope of the paper. It presents the main problems we consider, illustrating the relationships among them. Concretely, an arrow from  $A$  to  $B$  indicates that  $B$  is a special instance of  $A$ . It also divides the problems according to its difficulty, with and without bounded treewidth assumptions. In this paper we start from the simplest problems, i.e., permanents and determinants of matrices, moving upwards in the diagram.

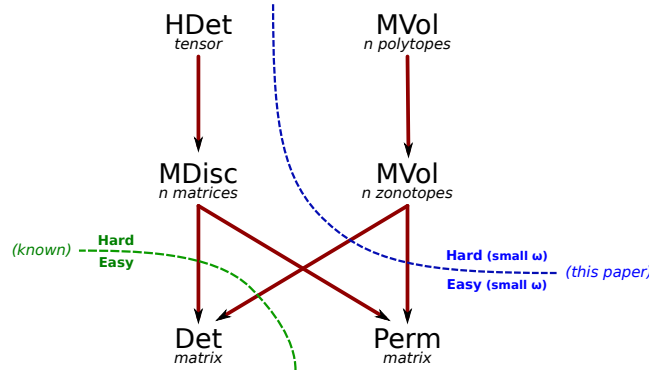


FIGURE 1. Diagram describing the complexity relations of computing: determinants, permanents, mixed discriminants, hyperdeterminants and mixed volumes.

The document is structured as follows. In Section 2 we review the concept of treewidth and tree decompositions. We also present three graph abstractions of a sparse matrix. Among these graphs is the bipartite graph  $G$  described above, and a projection  $G^X$  onto the column set. In Section 3 we present a decomposition method, Algorithm 1, that computes the permanent based on the graph  $G^X$ . We first use graph  $G^X$  because the decomposition algorithm is easier to explain in this case. In Section 4 we extend this method to work with the bipartite graph  $G$ , as shown in Algorithm 2.

We provide a Matlab implementation of this algorithm. In Section 5 we discuss the case of mixed discriminants, presenting a decomposition method for it. We also treat the case of generalized determinants/permanents on tensors. Finally, in Section 6 we discuss the case of mixed volumes of zonotopes.

## Related work.

*Permanents.* The best known to date method for exactly computing the permanent of a general matrix was given by Ryser and its complexity is  $O(n 2^n)$  [29]. There are two main research trends on permanent computation: approximation algorithms and exact algorithms for structured matrices. We briefly discuss related work in both of them. Tree decomposition algorithms, which belong to the second group, will be presented afterwards.

We first mention some work on approximation schemes. For arbitrary matrices, Gurvits gave a randomized polynomial time approximation, with error proportional to the operator norm [21]. For nonnegative matrices there is a vast literature, see e.g., [37] and the references therein. Most remarkably, Jerrum et al. gave a fully polynomial randomized approximation scheme (FPRAS) [23]. Recent work studies approximation schemes based on belief propagation, also for nonnegative matrices [37, 38].

As for exact algorithms for structured matrices, different types of structure have been explored in the literature. Fisher, Kasteleyn and Temperley gave a polynomial time algorithm for matrices whose associated bipartite graph is planar [24, 33]. Barvinok showed that the permanent is tractable when the rank is bounded [6]. The case of 0/1 circulant matrices has also been considered [27], as well as sparse 0/1 Toeplitz matrices [15]. Schwartz showed a  $O(\log(n)2^{6w})$  algorithm for certain band, Toeplitz matrices, where  $w$  is the bandwidth [31]. Temme and Wocjan showed a  $O(n 2^{3w^2})$  algorithm for a special type of band matrices [32]. Note that for arbitrary band matrices our algorithm is  $\tilde{O}(n 2^{2w})$ .

*Permanents and treewidth.* Tree decomposition methods for permanent computation have been considered. Courcelle et al. first showed that the permanent can be computed efficiently if the treewidth is bounded [16], although their methods, based on the Feferman-Vaught-Shelah Theorem, do not lead to an implementable algorithm. Later work of Flarup et al. gives a  $O(n 2^{O(w^2)})$  algorithm [20]. This algorithm is extended in [26] to a wider class of matrices. Note the strong dependency on the treewidth. Furthermore, the graph abstraction used in the above methods, which is not the bipartite graph  $G$ , has two inconvenient features: its treewidth can be significantly larger than the one of  $G$  (see Example 2.1) and it is dependent on the specific order of the columns of the matrix (see Remark 2.2).

Closer to this paper is the work of van Rooij et al. [36]. They gave a  $\tilde{O}(n 2^w)$  decomposition algorithm for counting perfect matchings in a graph. Counting perfect matchings is closely related to the permanent, and one could derive from their proof an analogous, but different, method for calculating the permanent. Our algorithm could be seen as a variant of such method that is easier to extend to the higher dimensional problems we consider.

*Mixed discriminants, mixed volumes, tensors.* The higher dimensional problems we study generalize the permanent of a matrix, and thus are #P-hard in general. As for the permanent, there are two natural relaxations: approximation algorithms and exact algorithms under special structure. Approximation algorithms have been considered for mixed discriminants [7, 21], mixed volumes [7, 19] and multidimensional permanents [8]. As for exact algorithms under special structure, we are only aware of Gurvits' tractability result for mixed discriminants and the 4-hyperdeterminant under some bounded rank assumptions [21].

To our knowledge this is the first paper that studies tree decomposition methods for mixed discriminants, mixed volumes and generalized determinants/permanents on tensors. Related to this is a recent log-space algorithm for computing determinants under bounded treewidth assumptions [3]. Also related is the problem of partitioning a low treewidth graph into  $k$ -cliques, which is considered in [36].

## 2. TREE DECOMPOSITIONS AND MATRIX GRAPHS

In this section we review some basic facts regarding tree decompositions of a graph. We also present three graphs that can be associated to a sparse matrix.

**2.1. Tree decompositions.** The notions of treewidth and tree decompositions are fundamental in many areas of computer science and applied mathematics [12, 17]. Intuitively, the treewidth of a graph  $G$  is a measure of how close it is to a tree. A graph has treewidth 1 if and only if it is a forest, i.e., a disjoint union of trees. The smaller the treewidth, the closer the graph is to a tree, and the easier it is to solve certain problems on it. We note that a graph of treewidth  $\omega$  has at most  $n\omega$  edges, and thus treewidth imposes a sparsity constraint. We give the formal definition now.

**Definition 2.1.** Let  $G$  be a graph with vertex set  $X$ . A *tree decomposition* of  $G$  is a pair  $(T, \chi)$ , where  $T$  is a rooted tree and  $\chi : T \rightarrow \{0, 1\}^X$  assigns some  $\chi(t) \subseteq X$  to each node  $t$  of  $T$ , that satisfies the following conditions.

- i. The union of  $\{\chi(t)\}_{t \in T}$  is the whole vertex set  $X$ .
- ii. For every edge  $(x_i, x_j)$  of  $G$ , there exists some node  $t$  of  $T$  with  $x_i, x_j \in \chi(t)$ .
- iii. For every  $x_i \in X$  the set  $\{t : x_i \in \chi(t)\}$  forms a subtree of  $T$ .

The sets  $\chi(t)$  are usually referred to as *bags*. The *width*  $\omega$  of the decomposition is the size of the largest bag (minus one). The *treewidth* of  $G$  is the minimum width among all possible tree decompositions.

Algorithms based on tree decompositions typically depend exponentially on the width of the decomposition, and polynomially on the number of nodes [12]. Thus, given a graph  $G$  it is desirable to obtain a tree decomposition of minimum width. However, finding the treewidth is NP-hard [1]. The treewidth of some simple graphs are known: for a tree is 1, for a cycle graph is 2, for the  $n \times n$  grid graph is  $n$ , for the complete graph  $K_n$  is  $n - 1$ , for the complete bipartite graph  $K_{n,n}$  is  $n$ . For general graphs, there are good heuristics and approximation algorithms [12].

Tree decompositions are closely related to chordal graphs [11, 17]. Indeed, given a chordal graph  $G$ , we can construct a tree decomposition where the bags  $\chi(t)$  correspond

to its maximal cliques. We remark that there are at most  $n$  maximal cliques in a chordal graph. In general, we can always assume that a tree decomposition has at most  $n$  nodes.

The following is a simple property of tree decompositions.

**Lemma 1.** *Let  $(T, \chi)$  be a tree decomposition of  $G$ . Then for any clique  $Y$  of  $G$  there is some node  $t$  of  $T$  with  $Y \subseteq \chi(t)$ .*

*Proof.* For each vertex  $y \in Y$ , let  $T_y$  denote the subtree of all bags containing  $y$ . Let  $t_y \in T_y$  be the closest node to the root and let  $d(t_y)$  denote the distance from  $t_y$  to the root. Observe that if  $d(t_y) \leq d(t_{y'})$  then  $t_{y'} \in T_y$  (otherwise,  $T_y \cap T_{y'} = \emptyset$  and the edge  $(y, y')$  would not belong to any bag). Let  $t \in \{t_y\}_{y \in Y}$  be the farthest away from the root, i.e.,  $d(t_y) \leq d(t)$  for all  $y \in Y$ . It follows that  $Y \subseteq \chi(t)$ .  $\square$

**2.2. Matrix graphs.** The sparsity structure of a matrix, i.e., its pattern of nonzero entries, can be described in terms of a graph. We consider here three possible graph abstractions of such sparsity structure, and we compare their treewidths.

Let  $M$  be a  $n \times n$  matrix. We will index the rows with a set  $A = \{a_1, \dots, a_n\}$  and the columns with a set  $X = \{x_1, \dots, x_n\}$ . We use subindices to index the coordinates of  $M$ , i.e.,  $M_{a,x}$  denotes the entry in the  $a$ -th row and  $x$ -th column. Similarly, let  $M_a$  be the  $a$ -th row of  $M$ . We now present two (undirected) graphs that are usually associated to a sparse matrix.

**Definition 2.2** (Bipartite graph). Let  $M$  be a  $n \times n$  matrix, let  $A$  denote its set of rows, and let  $X$  denote its set of columns. The *bipartite graph* of  $M$ , denoted as  $G(M)$ , has vertices  $A \cup X$ , and there is an edge  $(a, x)$  if  $M_{a,x}$  is nonzero.

**Definition 2.3.** (Symmetrized graph) Let  $M$  be a  $n \times n$  matrix, let  $A$  denote its set of rows, and let  $X$  denote its set of columns. The *symmetrized graph* of  $M$ , denoted as  $G^s(M)$ , has vertices  $1, \dots, n$  and has an edge  $(i, j)$  if  $M_{a_i, x_j} \neq 0$  or  $M_{a_j, x_i} \neq 0$ .

*Remark 2.1.* Note that  $G^s(M)$  is the adjacency graph of the symmetric matrix  $M + M^T$ , assuming no terms cancel out.

*Remark 2.2* (Permutation invariance). Note that the permanent of a matrix is invariant under independent row and column permutations. The bipartite graph  $G$  preserves this invariance. On the other hand, the symmetrized graph  $G^s$  is only invariant under simultaneous row and column permutations.

The bipartite graph  $G$  is our main object of study in this paper. Let  $\omega := \text{tw}(G)$  be the treewidth of  $G$  and  $\omega_s := \text{tw}(G^s)$  the one of  $G^s$ . Tree decomposition methods based on graph  $G^s$  have been studied before [3, 16, 20, 26]. We claim that graph  $G$  is a better abstraction for the purpose of permanent computation. In particular,  $G$  preserves the permutation invariance of the permanent as stated above. Furthermore,  $\omega_s$  can be much larger than  $\omega$  as shown in the following example.

**Example 2.1** (Two nonzero entries per row). Let  $M$  be a matrix with at most two nonzero entries per row. We claim that for all nontrivial cases the bipartite graph  $G$  has treewidth  $\omega \leq 2$ . Let  $G_0$  be a connected component, and let  $n_0$  be its number of row vertices. In order for  $G_0$  to have a perfect matching, it must have as many row

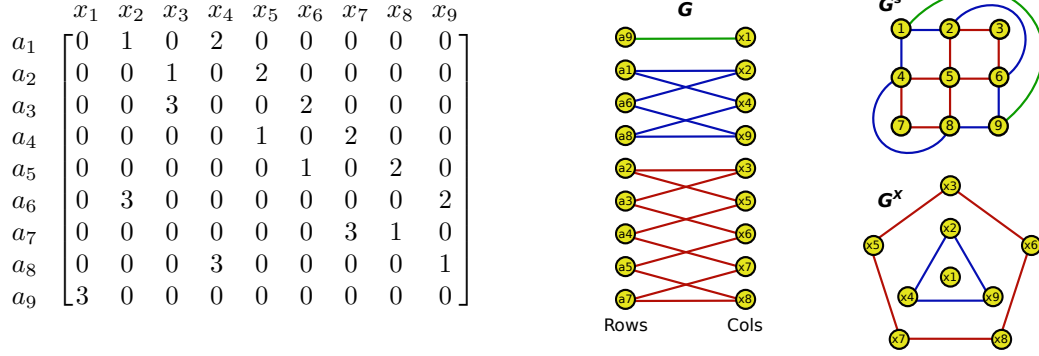


FIGURE 2. Graph abstractions of a matrix: bipartite graph  $G$ , symmetrized graph  $G^s$  and column graph  $G^X$ .

vertices as column vertices. Note also that  $G_0$  has at most  $2n_0$  edges because the row degrees are at most 2. Then  $G_0$  is a connected graph with  $2n_0$  vertices and at most  $2n_0$  edges, so it has at most one cycle. It follows that  $\omega \leq 2$ .

On the other hand, we will see that  $\omega_s$  is unbounded. Let  $n = m^2$  and consider the matrix  $M$  whose nonzero entries are

$$\begin{aligned}
 M_{a_i, x_{i+1}} &= 1, & \text{if } m \text{ does not divide } i \\
 M_{a_i, x_{i+m}} &= 2, & \text{if } 1 \leq i \leq n - m \\
 M_{a_{(m-i+1)m}, x_i} &= 3, & \text{if } 1 \leq i \leq m \\
 M_{a_{n-i}, x_{im+1}} &= 3, & \text{if } 1 \leq i < m
 \end{aligned}$$

The corresponding graph  $G^s$  contains the grid graph, and thus  $\omega_s \geq \sqrt{n}$ . The case  $n = 9$  is shown in Figure 2.

The following example shows that the treewidth of  $G$  is always better than the treewidth of  $G^s$ .

**Example 2.2** ( $G$  is “better” than  $G^s$ ). Let’s see that given a tree decomposition of  $G^s$  of width  $\omega_s$  we can form a tree decomposition of  $G$  of width  $2\omega_s$ . Let  $(T, \iota)$  be a tree decomposition of  $G^s$ , where  $\iota(t) \subseteq \{1, \dots, n\}$ . Let  $\mu : T \rightarrow \{0, 1\}^{A \cup X}$ , be such that  $\mu(t) = \{a_i : i \in \iota(t)\} \cup \{x_i : i \in \iota(t)\}$ . Then  $(T, \mu)$  is a decomposition of  $G$  of width  $2\omega_s$ . On the contrary, for a fixed  $\omega$  the treewidth of  $G^s$  is unbounded as seen in Example 2.1.

We now introduce a third graph  $G^X$  that we can associate to matrix  $M$ .

**Definition 2.4.** (Column graph) Let  $M$  be a  $n \times n$  matrix, let  $A$  denote its set of rows and let  $X$  denote its set of columns. For any  $a \in A$  let  $X(a)$  denote the set of nonzero components of row  $M_a$ . The *column graph*  $G^X(M)$  has  $X$  as its vertex set, and for each  $a \in A$  we put a clique in  $X(a)$ . Equivalently, there is an edge  $(x_i, x_j)$  if there is some  $a \in A$  such that  $x_i, x_j \in X(a)$ .

Graph  $G^X$  can be seen as a projection of  $G$  onto the column set  $X$ . We show in the following example that  $\omega \leq \omega_X + 1$ , where  $\omega_X := \text{tw}(G^X)$ .

**Example 2.3** ( $G$  is “better” than  $G^X$ ). Let  $(T, \chi)$  be a tree decomposition of  $G^X$  of width  $\omega_X$ . For each row  $a \in A$  we associate to it a unique node  $t \in T$  such that  $X(a) \subseteq \chi(t)$ . This assignment can be made because of Lemma 1. For some  $t \in T$ , let  $a_1^t, a_2^t, \dots, a_k^t$  be all rows that are assigned to  $t$ . Let’s replace node  $t$  of  $T$  with a path  $t_1, t_2, \dots, t_k$ , and let  $\mu(t_j) = \chi(t) \cup \{a_j^t\}$  for  $j = 1, \dots, k$ . The nodes previously connected to  $t$  can be linked to any of the new nodes. By doing this for every  $t \in T$ , we obtain a tree decomposition  $(T, \mu)$  of  $G$  of width  $\omega_X + 1$ .

On the other hand, for a fixed  $\omega$  the treewidth of  $G^X$  is unbounded. For instance, consider the matrix  $M$  whose only nonzero entries are:  $M_{a_i, x_i} = 1$  (diagonal) and  $M_{a_1, x_i} = 1$  (first row) for all  $i$ . In this case  $G^X$  is the complete graph ( $\omega_X = n - 1$ ) and  $G$  is a tree ( $\omega = 1$ ).

The reason why we consider the graph  $G^X$  is that we can give a very simple algorithm for the permanent based on it. We present this algorithm in Section 3, and then extend it to  $G$  in Section 4.

To conclude this section, let’s see how the three graphs  $G, G^s, G^X$  can capture the special case of a band matrix.

**Example 2.4** (Band matrix). Let  $w_1, w_2 \in \mathbb{Z}_{>0}$  and let  $M$  be such that  $M_{a_i, x_j} = 0$  if either  $i - j > w_1$  or  $j - i > w_2$ . Let  $T$  be a path  $t_1, t_2, t_3, \dots$ . Optimal decompositions  $(T, \chi), (T, \iota), (T, \mu)$  for the graphs  $G^X, G^s, G$  respectively, are given by

$$\begin{aligned} \chi(t_i) &:= \{x_i, x_{i+1}, \dots, x_{i+w_1+w_2}\}, & \iota(t_i) &:= \{i, i+1, \dots, i + \max\{w_1, w_2\}\}, \\ \mu(t_{2i-1}) &:= \{a_{i+w_1}, x_i, x_{i+1}, \dots, x_{i+w_1+w_2-1}\}, & \mu(t_{2i}) &:= \{a_{i+w_1}, x_{i+1}, x_{i+1}, \dots, x_{i+w_1+w_2}\} \end{aligned}$$

The widths of these decompositions are  $\omega = \omega_X = w_1 + w_2$  and  $\omega_s = \max\{w_1, w_2\}$ .

### 3. COLUMN DECOMPOSITIONS

**Notation.** For sets  $Y, Y_1, Y_2$ , let  $Y = Y_1 \sqcup Y_2$  denote a set partition, i.e.,  $Y = Y_1 \cup Y_2$  and  $Y_1 \cap Y_2 = \emptyset$ .

In this section, we develop an algorithm to compute the permanent based on a tree decomposition of the column graph  $G^X$  (see Definition 2.4). For this section only, we denote the treewidth of  $G^X$  by  $\omega$ . We will show that we can compute  $\text{Perm}(M)$  in  $\tilde{O}(n 4^\omega)$ . Recall that the notation  $\tilde{O}$  ignores polynomial factors in  $\omega$ .

As before,  $A$  denotes the row set and  $X$  the column set. We use subindices to index the coordinates of  $M$ , i.e.,  $M_{a,x}$  denotes the element in row  $a$  and column  $x$ . The following example illustrates the methodology we follow.

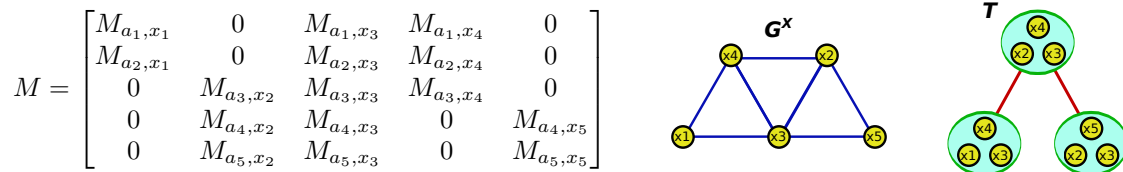


FIGURE 3. Matrix  $M$ , its column graph  $G^X$  and a tree decomposition  $T$ .

**Example 3.1.** Consider the  $5 \times 5$  matrix  $M$  of Figure 3 and the following partition of its rows:

$$A = \{a_1, a_2\} \sqcup \{a_3\} \sqcup \{a_4, a_5\}.$$

There is a simple expansion of  $\text{Perm}(M)$  in terms of this partition:

$$\begin{aligned} \text{Perm}(M) = & \text{Perm} \left( \begin{bmatrix} M_{a_1,x_1} & M_{a_1,x_3} \\ M_{a_2,x_1} & M_{a_2,x_3} \end{bmatrix} \right) \text{Perm}([M_{a_3,x_4}]) \text{Perm} \left( \begin{bmatrix} M_{a_4,x_2} & M_{a_4,x_5} \\ M_{a_5,x_2} & M_{a_5,x_5} \end{bmatrix} \right) \\ & + \text{Perm} \left( \begin{bmatrix} M_{a_1,x_1} & M_{a_1,x_4} \\ M_{a_2,x_1} & M_{a_2,x_4} \end{bmatrix} \right) \text{Perm}([M_{a_3,x_3}]) \text{Perm} \left( \begin{bmatrix} M_{a_4,x_2} & M_{a_4,x_5} \\ M_{a_5,x_2} & M_{a_5,x_5} \end{bmatrix} \right) \\ & + \text{Perm} \left( \begin{bmatrix} M_{a_1,x_1} & M_{a_1,x_4} \\ M_{a_2,x_1} & M_{a_2,x_4} \end{bmatrix} \right) \text{Perm}([M_{a_3,x_2}]) \text{Perm} \left( \begin{bmatrix} M_{a_4,x_3} & M_{a_4,x_5} \\ M_{a_5,x_3} & M_{a_5,x_5} \end{bmatrix} \right). \end{aligned}$$

This expansion implies that to compute  $\text{Perm}(M)$  we just need to evaluate two  $2 \times 2$  permanents corresponding to  $\{a_1, a_2\}$ , three  $1 \times 1$  permanents corresponding to  $\{a_3\}$ , and two  $2 \times 2$  permanents corresponding to  $\{a_4, a_5\}$ . This requires only 14 multiplications, compared to  $4 \times 5! = 480$  multiplications using the definition. The reason why this formula exists is because the column graph  $G^X$  of matrix  $M$  has a simple tree decomposition, which is shown in Figure 3.

As in the example above, we can always obtain an expansion of  $\text{Perm}(M)$  using a tree decomposition of graph  $G^X$ . By carefully evaluating this formula we will obtain a dynamic programming method to compute  $\text{Perm}(M)$ .

**3.1. Partial permanent.** In our notation, the permanent of  $M$  can be expressed as

$$\text{Perm}(M) = \sum_{\pi} \prod_{a \in A} M_{a,\pi(a)}$$

where the sum is over all bijections  $\pi : A \rightarrow X$ . For a given row  $a$ , let  $X(a)$  denote the column coordinates where it is nonzero. We will refer to a bijection  $\pi$  as a *matching* if  $\pi(a) \in X(a)$ , i.e.,  $M_{a,\pi(a)} \neq 0$ , for all  $a \in A$ . Then we can restrict the above sum to be over all matchings.

We consider a tree decomposition  $(T, \chi)$  of the column graph  $G^X$ . Note that by construction of  $G^X$  then  $X(a)$  is a clique for any  $a \in A$ . Thus, Lemma 1 says that we can assign each row  $a \in A$  to some node  $t$ , such that  $X(a) \subseteq \chi(t)$ . From now, we *fix an assignment* of each  $a$  to a unique node. Let  $A_t$  denote the rows that are assigned to node  $t$ . Thus, we have the partition  $A = \bigsqcup_{t \in T} A_t$ .

For a node  $t \in T$ , we denote the subtree of  $T$  rooted in  $t$  by  $T_t$ . Let  $\chi(T_t)$  be the union of  $\chi(t')$  over all  $t' \in T_t$ , and similarly let  $A_{T_t}$  be the union of  $A_{t'}$  over all  $t' \in T_t$ . For instance, for the root node we have  $A_{T_{\text{root}}} = A$  and  $\chi(T_{\text{root}}) = X$ .

For a fixed matrix  $M$  and some sets  $D \subseteq A$  and  $Y \subseteq X$  we denote

$$(1) \quad \text{perm}(D, Y) := \sum_{\pi} \prod_{a \in D} M_{a,\pi(a)}$$

where the sum is over all matchings  $\pi : D \rightarrow Y$ . Equivalently, it is the permanent of the submatrix of  $M$  corresponding to such rows and columns. Clearly, this only



makes sense if  $|D| = |Y|$ , and otherwise we can define  $\text{perm}(D, Y) = 0$ . We refer to  $\text{perm}(D, Y)$  as a *partial permanent*.

**3.2. Decomposition algorithm for the permanent.** Algorithm 1 presents our dynamic programming method to compute  $\text{Perm}(M)$ . We will explain and derive this algorithm in the following sections. For each node  $t$  of the tree, the algorithm computes a table  $P_t$  indexed by subsets  $\bar{Y}$  of  $\chi(t)$ . It starts from the leaves of the tree and recursively computes the tables of all nodes following a topological ordering. The permanent of  $M$  is found in the table corresponding to the root.

---

**Algorithm 1** Permanent with column decomposition

---

**Input:** Matrix  $M$  and tree decomposition  $(T, \chi)$  of column graph  $G^X(M)$

**Output:** Permanent of  $M$

```

1: procedure COLSPERM( $M, T, \chi$ )
2:   assign each  $a \in A$  to some  $t$  with  $X(a) \subseteq \chi(t)$ 
3:    $order :=$  topological ordering of  $T$  starting from its leaves
4:   for  $t$  in  $order$  do
5:      $Q_t := \text{SUBPERMS}(t, M)$ 
6:     if  $t$  is a leaf then
7:        $P_t := Q_t$ 
8:     else
9:        $c_1, \dots, c_k :=$  children of  $t$ 
10:       $P_t := \text{EVALRECURSION}(t, Q_t, P_{c_1}, \dots, P_{c_k})$ 
11:   return  $P_{root}(\chi(root))$ 
12: procedure SUBPERMS( $t, M$ )
13:    $A_t :=$  rows assigned to node  $t$ 
14:    $Q_t(\bar{Y}) := \text{perm}(A_t, \bar{Y})$  for all  $\bar{Y} \subseteq \chi(t)$ 
15: procedure EVALRECURSION( $t, Q_t, P_{c_1}, \dots, P_{c_k}$ )
16:   for  $c_j$  child of  $t$  do
17:      $\Delta_j := \chi(c_j) \setminus \chi(t), \quad \Lambda_j := \chi(c_j) \cap \chi(t)$ 
18:      $Q_{c_j}(\bar{Y}) := P_{c_j}(\bar{Y} \cup \Delta_j)$  for all  $\bar{Y} \subseteq \Lambda_j$ 
19:    $P_t := \text{SUBSETCONVOLUTION}(Q_t, Q_{c_1}, \dots, Q_{c_k})$ 
20: procedure SUBSETCONVOLUTION( $P_0, P_1, \dots, P_k$ )
21:    $P(\bar{Y}) := \sum_{\bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_k = \bar{Y}} P_0(\bar{Y}_0) P_1(\bar{Y}_1) \dots P_k(\bar{Y}_k)$ 

```

---

Algorithm 1 has two main routines:

- For any node  $t$ , SUBPERMS computes a table  $Q_t$  with the permanents of all submatrices corresponding to  $t$ . (See Section 3.3).
- EVALRECURSION computes table  $P_t$  of an internal node  $t$ , by combining table  $Q_t$  with the tables  $P_{c_1}, \dots, P_{c_k}$  of the node's children. (See Section 3.4).

The values  $P_t(\bar{Y})$  that we compute correspond to a partial permanent of the matrix, as we explain now. Consider the collection

$$S := \{Y : \chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t)\}.$$

Observe that  $Y \in S$  is completely determined by  $Y \cap \chi(t)$ . Therefore, if we let  $\bar{Y} := Y \cap \chi(t)$ , there is a one to one correspondence between  $S$ , and the collection  $\bar{S} := \{\bar{Y} : \bar{Y} \subseteq \chi(t)\}$ . Then the partial permanents that we are interested in are

$$P_t(\bar{Y}) := \text{perm}(A_{T_t}, Y) = \text{perm}(A_{T_t}, \bar{Y} \cup (\chi(T_t) \setminus \chi(t))).$$

The reason why we index table  $P_t$  with  $\bar{Y}$  instead of  $Y$ , is that in this way it becomes clearer that the recursion formula is actually a subset convolution.

Observe that the permanent of  $M$  is indeed computed in Algorithm 1, as for the root node we have  $P_{\text{root}}(\chi(\text{root})) = \text{perm}(A, X) = \text{Perm}(M)$ . Also note that for a leaf node we have  $P_t(\bar{Y}) = \text{perm}(A_t, \bar{Y}) = Q_t(\bar{Y})$ .

**Example 3.2.** Consider the matrix  $M$  and tree decomposition  $T$  of Figure 3. Let  $t_1, t_2, t_3$  be the nodes of  $T$ , where the central node  $t_2$  is the root. We show the tables computed by Algorithm 1. The tables  $Q_t$  with the permanents of all submatrices are:

$$\begin{aligned} Q_{t_1}(\{x, y\}) &= \text{perm}(\{a_1, a_2\}, \{x, y\}), & \text{for } x, y \in \chi(t_1) = \{x_1, x_3, x_4\} \\ Q_{t_3}(\{x, y\}) &= \text{perm}(\{a_4, a_5\}, \{x, y\}), & \text{for } x, y \in \chi(t_3) = \{x_2, x_3, x_5\} \\ Q_{t_2}(\{x\}) &= \text{perm}(\{a_3\}, \{x\}), & \text{for } x \in \chi(t_2) = \{x_2, x_3, x_4\} \end{aligned}$$

We now show the final tables  $P_t$  for each node. For the leaves  $t_1, t_3$  we have  $P_{t_1} = Q_{t_1}$ ,  $P_{t_3} = Q_{t_3}$ . As for the root  $t_2$ , the recursion is:

$$\begin{aligned} P_{t_2}(\{x_2, x_3, x_4\}) &= Q_{t_2}(\{x_4\})P_{t_1}(\{x_1, x_3\})P_{t_3}(\{x_2, x_5\}) + \\ &\quad Q_{t_2}(\{x_3\})P_{t_1}(\{x_1, x_4\})P_{t_3}(\{x_2, x_5\}) + Q_{t_2}(\{x_2\})P_{t_1}(\{x_1, x_4\})P_{t_3}(\{x_3, x_5\}). \end{aligned}$$

Note that this recursion matches the permanent expansion in Example 3.1.

In the following sections we explain the two main routines of Algorithm 1, i.e., SUBPERMS and EVALRECURSION, obtaining complexity bounds for them.

**3.3. Permanent of all submatrices.** Let  $M_0$  be a rectangular matrix with row set  $A_0$  and column set  $X_0$ . As a part of our algorithm, which can be seen as the base case, we require a good method to compute the permanents of all submatrices of  $M_0$ . In other words, we want to obtain the partial permanents  $\text{perm}(D, Y)$  for all pairs  $(D, Y)$ . We can do this in a very simple way using an expansion by minors. The following lemma explains such procedure and gives its running time.

**Lemma 2.** *Let  $M_0$  be a matrix of dimensions  $n_1 \times n_2$ . Let  $A_0$  denote its row set,  $X_0$  its column set and let  $S = \{(D, Y) \subseteq A_0 \times X_0 : |D| = |Y|\}$ . We can compute  $\text{perm}(D, Y)$  for all  $(D, Y) \in S$  in  $O(n_{\max}^2 2^{n_1+n_2})$ , where  $n_{\max} = \max\{n_1, n_2\}$ .*

*Proof.* Let  $S_i = \{(D, Y) : |D| = |Y| = i\}$  for  $1 \leq i \leq \min\{n_1, n_2\}$ . We use an expansion by minors to compute  $\text{perm}(D, Y)$  for  $(D, Y) \in S_i$ , using the values of  $S_{i-1}$ . Let  $a_0$  be

the first element in  $D$ , then

$$\text{perm}(D, Y) = \sum_{x \in Y} M_{a_0, x} \text{perm}(D \setminus a_0, Y \setminus x).$$

Thus, for each  $(D, Y)$ , we loop over at most  $n_2$  elements, and for each we need  $O(n_{\max})$  to find the sets  $D \setminus a_0$  and  $Y \setminus x$ . The result follows.  $\square$

**3.4. Recursion formula.** The heart of Algorithm 1 is given by the recursion formula used, i.e., the procedure to obtain table  $P_t$  of node  $t$  from the tables of its children. This recursion formula is given in the following lemma.

**Lemma 3.** *Let  $M$  be a matrix with associated column graph  $G^X$ . Let  $(T, \chi)$  be a tree decomposition of  $G^X$ . Let  $t$  be an internal node of  $T$ , and let  $Y$  be such that*

$$(2) \quad \chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t), \quad |Y| = |A_{T_t}|$$

*Let  $c_1, \dots, c_k$  be the children of  $t$ . Then*

$$(3) \quad \text{perm}(A_{T_t}, Y) = \sum_{\mathcal{Y}} \text{perm}(A_t, Y_t) \prod_{j=1}^k \text{perm}(A_{T_{c_j}}, Y_{c_j})$$

*where  $\text{perm}(\cdot, \cdot)$  is as in (1) and the sum is over all  $\mathcal{Y} = (Y_t, Y_{c_1}, \dots, Y_{c_k})$  such that:*

$$(4a) \quad Y = Y_t \sqcup (Y_{c_1} \sqcup \dots \sqcup Y_{c_k})$$

$$(4b) \quad \chi(T_{c_j}) \setminus \chi(t) \subseteq Y_{c_j} \subseteq \chi(T_{c_j}) \quad Y_t \subseteq \chi(t).$$

*Proof.* Observe that  $A_{T_t}$  can be partitioned as

$$A_{T_t} = A_t \sqcup (A_{T_{c_1}} \sqcup \dots \sqcup A_{T_{c_k}}).$$

Let  $\pi : A_{T_t} \rightarrow Y$  be a matching. Let  $c$  be a child of  $t$  and let  $\pi_c : A_{T_c} \rightarrow Y$  be the restriction of  $\pi$  to  $A_{T_c}$ . Let  $Y_t := \pi(A_t) \subseteq \chi(t)$  be the range of  $\pi$  restricted to  $A_t$ , and let  $Y_c$  be the range of  $\pi_c$ . As  $\pi$  is injective, then equation (4a) holds. Observe also that  $Y_c = \pi(A_{T_c}) \subseteq \chi(T_c)$ . Note now that if  $x \in \chi(T_c) \setminus \chi(t)$ , then it is in the range of  $\pi$ . However, as  $x \notin \chi(t)$  then  $x \notin \chi(T_{c'})$  for any other child  $c'$ , and thus  $x$  has to be in the range of  $\pi_c$ . Thus, the range of  $\pi_c$ , i.e.,  $Y_c$ , contains  $\chi(T_c) \setminus \chi(t)$ .

Therefore, for any matching  $\pi : A_{T_t} \rightarrow Y$  and for any child  $c$ ,  $\pi$  induces a matching from  $A_{T_c}$  to some  $Y_c$  that satisfy equations (4). On the other hand, given  $Y_t, Y_{c_1}, \dots$  satisfying (4) and matchings  $\pi_t, \pi_{c_1}, \dots$  on  $A_t, A_{T_{c_1}}, \dots$  with such ranges, we can merge them into a function on  $A_{T_t}$ . Observe that (4) ensures that the ranges of these matchings are disjoint and their union is  $Y$ . We conclude that

$$\begin{aligned} \text{perm}(A_{T_t}, Y) &= \sum_{\pi: A_{T_t} \rightarrow Y} \prod_a M_{a, \pi(a)} \\ &= \sum_{\mathcal{Y}} \sum_{\substack{\pi_t: A_t \rightarrow Y_t \\ \pi_c: A_{T_c} \rightarrow Y_c}} \left( \prod_{a_t} M_{a_t, \pi_t(a_t)} \right) \prod_{c_j} \left( \prod_{a_c} M_{a_c, \pi_{c_j}(a_c)} \right) \\ &= \sum_{\mathcal{Y}} \text{perm}(A_t, Y_t) \prod_{c_j} \text{perm}(A_{T_{c_j}}, Y_{c_j}). \end{aligned}$$

□

At first sight, the recursion of equation (3) looks difficult to evaluate. It turns out that this formula is a subset convolution and thus it can be computed efficiently using the algorithm from [10], as explained in the following lemma. We follow this approach in method EVALRECURSION of Algorithm 1.

**Lemma 4.** *Given the values of the partial permanents  $\text{perm}(A_t, Y_t)$  and  $\text{perm}(A_{T_{c_j}}, Y_{c_j})$ , we can evaluate equation (3) for all  $Y$  satisfying (2) in  $\tilde{O}(k 2^\omega)$ .*

*Proof.* Let  $\bar{Y} := Y \cap \chi(t)$ ,  $\bar{Y}_t := Y_t$ ,  $\bar{Y}_{c_j} := Y_{c_j} \cap \chi(t)$ , and let

$$\Delta_t := \chi(T_t) \setminus \chi(t) \quad \Delta_{c_j} := \chi(T_{c_j}) \setminus \chi(t)$$

$$P_t(\bar{Y}) := \text{perm}(A_{T_t}, \bar{Y} \cup \Delta_t), \quad Q_t(\bar{Y}_t) := \text{perm}(A_t, \bar{Y}_t), \quad Q_{c_j}(\bar{Y}_{c_j}) := \text{perm}(A_{T_{c_j}}, \bar{Y}_{c_j} \cup \Delta_{c_j})$$

Then equation (3) can be rewritten as

$$(5) \quad P_t(\bar{Y}) = \sum_{\bar{Y}_t \sqcup \bar{Y}_{c_1} \sqcup \dots \sqcup \bar{Y}_{c_k} = \bar{Y}} Q_t(\bar{Y}_t) \prod_{j=1}^k Q_{c_j}(\bar{Y}_{c_j})$$

where  $\bar{Y}_t \subseteq \chi(t)$  and  $\bar{Y}_{c_j} \subseteq \chi(c_j) \cap \chi(t)$ . Equation (5) is a subset convolution over the subsets of  $\chi(t)$ . Therefore, it can be evaluated in  $O(kw^2 2^w)$ , where  $w = |\chi(t)|$ , using the algorithm from [10]. □

The following theorem gives the running time of Algorithm 1, proving that we can efficiently compute the permanent given a tree decomposition of  $G^X$  of small width.

**Theorem 5.** *Let  $M$  be a matrix with associated column graph  $G^X$ . Let  $(T, \chi)$  be a tree decomposition of  $G^X$  of width  $\omega$ . Then we can compute  $\text{Perm}(M)$  in  $\tilde{O}(n 4^\omega)$ .*

*Proof.* Let  $t$  be some node in  $T$ . We compute  $\text{perm}(A_{T_t}, Y)$  for every  $Y$  satisfying (2). In particular, we will obtain  $\text{Perm}(A) = \text{perm}(A_{T_{\text{root}}}, \chi(T_{\text{root}}))$ . We will show that for each  $t$  we can compute  $\text{perm}(A_{T_t}, Y)$  for all  $Y$  in  $\tilde{O}((k_t + 1)4^\omega)$ , where  $k_t$  is the number of children of  $t$ . Note that  $\sum_t k_t$  is the number of nodes of tree  $T$ . As the tree has  $O(n)$  nodes, the total cost is then  $\tilde{O}(n 4^\omega)$ , as wanted.

The base case is when  $t$  is a leaf of  $T$ , so that  $A_{T_t} = A_t$  and  $\chi(T_t) = \chi(t)$ . Let  $M_0$  be the submatrix of  $M$  with rows  $A_t$  and columns  $\chi(t)$ . Then all we have to do is to obtain the permanent of some submatrices of  $M_0$ . Observe that  $|A_t| \leq |\chi(t)| \leq \omega$ , as otherwise there is no  $Y$  satisfying (2). Thus, we can do this in  $\tilde{O}(2^{2\omega})$  using Lemma 2.

Assume now that  $t$  is an internal node of  $T$  with  $k_t$  children and let  $Y$  that satisfies (2). Then equation (3) tells us how to find  $\text{perm}(A_{T_t}, Y)$ . Lemma 4 says that we can evaluate the formula in  $\tilde{O}(k_t 2^\omega)$ , assuming we know the values of the terms in the recursion. Note that we already found  $\text{perm}(A_{T_{c_j}}, Y_{c_j})$  for all children and that we can find  $\text{perm}(A_t, Y_t)$  for all possible  $Y_t$  in  $\tilde{O}(2^{2\omega})$  in the same way as for the base case. Thus, it takes  $\tilde{O}(4^\omega + k_t 2^\omega) = \tilde{O}((k_t + 1)4^\omega)$  to compute  $\text{perm}(A_{T_t}, Y)$  for all  $Y$ . □

*Remark 3.1.* The factor  $\tilde{O}(4^\omega)$  in the proof came from the base case. This bound can be improved, but we omit this as for the approach of Section 4, based on the bipartite graph, the bound will be  $\tilde{O}(2^\omega)$ .

**3.5. Computing the determinant.** Given the similarity between permanent and determinant, it should be possible to find an analogous algorithm for the determinant. We will derive such algorithm in this section. Ironically, this algorithm is slower than the one for the permanent. The reason is that the approach we follow does not take advantage of linear algebra: we loop over all permutations (carefully) and then compute its sign. We remark that our algorithm does not use divisions and thus can be applied in any commutative ring. The ideas from this section will be used in Section 5 to derive a decomposition algorithm for the mixed discriminant.

**Example 3.3.** Consider again the matrix  $M$  of Figure 3, and observe that a similar expansion holds for the determinant:

$$\begin{aligned} \text{Det}(M) = & \text{Det} \left( \begin{bmatrix} M_{a_1,x_1} & M_{a_1,x_3} \\ M_{a_2,x_1} & M_{a_2,x_3} \end{bmatrix} \right) \text{Det}([M_{a_3,x_4}]) \text{Det} \left( \begin{bmatrix} M_{a_4,x_2} & M_{a_4,x_5} \\ M_{a_5,x_2} & M_{a_5,x_5} \end{bmatrix} \right) \\ & - \text{Det} \left( \begin{bmatrix} M_{a_1,x_1} & M_{a_1,x_4} \\ M_{a_2,x_1} & M_{a_2,x_4} \end{bmatrix} \right) \text{Det}([M_{a_3,x_3}]) \text{Det} \left( \begin{bmatrix} M_{a_4,x_2} & M_{a_4,x_5} \\ M_{a_5,x_2} & M_{a_5,x_5} \end{bmatrix} \right) \\ & + \text{Det} \left( \begin{bmatrix} M_{a_1,x_1} & M_{a_1,x_4} \\ M_{a_2,x_1} & M_{a_2,x_4} \end{bmatrix} \right) \text{Det}([M_{a_3,x_2}]) \text{Det} \left( \begin{bmatrix} M_{a_4,x_3} & M_{a_4,x_5} \\ M_{a_5,x_3} & M_{a_5,x_5} \end{bmatrix} \right). \end{aligned}$$

As suggested in the above formula, the recursion used to compute the permanent can also be used to compute the determinant, by appropriately selecting the signs.

We recall now the definition of the parity function, and we extend it to ordered partitions.

**Definition 3.1.** Let  $D, Y$  be ordered sets of the same size. For a bijection  $\pi : D \rightarrow Y$  we define its *sign* or *parity* as  $\text{sgn}(\pi) := (-1)^{N(\pi)}$ , where  $N(\pi)$  is its number of inversions:

$$N(\pi) := |\{(a, a') \in D^2 : a < a', \pi(a) > \pi(a')\}|.$$

Let  $\mathcal{Y} = (Y_1, \dots, Y_k)$  be an *ordered partition* of  $Y$ , i.e.,  $Y = Y_1 \sqcup \dots \sqcup Y_k$ . We define its *sign* to be  $\text{sgn}(\mathcal{Y}) := (-1)^{N(\mathcal{Y})}$ , where  $N(\mathcal{Y})$  is:

$$N(\mathcal{Y}) := |\{(y_i, y_j) : y_i \in Y_i, y_j \in Y_j, i < j, y_i > y_j\}|.$$

Equivalently, we can associate to  $\mathcal{Y}$  a permutation  $\pi^{\mathcal{Y}} : \{1, 2, \dots, |Y|\} \rightarrow Y$  that consists of blocks: we put first  $Y_1$  (sorted), then  $Y_2$  (sorted), and so on. Then  $\text{sgn}(\mathcal{Y}) = \text{sgn}(\pi^{\mathcal{Y}})$ .

From the definition above it is clear that we can obtain the sign of a permutation in  $O(n^2)$  by counting the number of inversions. However, it is well known that we can find it in  $O(n)$  by counting its cycles.

For a matrix  $M$ , there is a natural order for its row set  $A$  and column set  $X$ , namely from top to bottom and from left to right. We recall the definition of the determinant:

$$\text{Det}(M) = \sum_{\pi} \text{sgn}(\pi) \prod_{a \in A} M_{a, \pi(a)}$$

where the sum is over all bijections  $\pi : A \rightarrow X$ . Similarly, for a fixed matrix  $M$  and for some  $D \subseteq A$  and  $Y \subseteq X$  we define the partial determinants:

$$(6) \quad \det(D, Y) := \sum_{\pi} \operatorname{sgn}(\pi) \prod_{a \in D} M_{a, \pi(a)}$$

where the sum is over all bijections  $\pi : D \rightarrow Y$ . Note that  $\operatorname{Det}(M) = \det(A, X)$ .

We now provide a recursion formula similar to the one in Lemma 3. We need one lemma before.

**Lemma 6.** *Let  $D, Y$  be ordered sets, and let  $\pi : D \rightarrow Y$  be a bijection, which we view as a subset of  $D \times Y$ . Let  $\mathcal{D} = (D_1, \dots, D_k)$  and  $\mathcal{Y} = (Y_1, \dots, Y_k)$  be partitions of  $D$  and  $Y$ . Let  $\pi = \pi_1 \sqcup \dots \sqcup \pi_k$  be a decomposition with  $\pi_j \subseteq D_j \times Y_j$ . Then*

$$\operatorname{sgn}(\pi) = \operatorname{sgn}(\mathcal{D}) \operatorname{sgn}(\mathcal{Y}) \prod_{j=1}^k \operatorname{sgn}(\pi_j).$$

*Proof.* It follows from the multiplicativity of the sign function.  $\square$

**Lemma 7.** *Under the same conditions of Lemma 3, then*

$$(7) \quad \det(A_{T_t}, Y) = \operatorname{sgn}(\mathcal{D}) \sum_{\mathcal{Y}} \operatorname{sgn}(\mathcal{Y}) \det(A_t, Y_t) \prod_{j=1}^k \det(A_{T_{c_j}}, Y_{c_j})$$

where  $\det(\cdot, \cdot)$  is as in (6) the sum is over all  $\mathcal{Y} = (Y_s, Y_{c_1}, \dots, Y_{c_k})$  satisfying (4),  $\mathcal{D} = (A_t, A_{T_{c_1}}, \dots, A_{T_{c_k}})$  and  $\operatorname{sgn}(\cdot)$  is as in Definition 3.1.

*Proof.* The proof is basically the same as the one of Lemma 3. The only difference is that we have the additional factor  $\operatorname{sgn}(\pi)$ , but it factors because of Lemma 6.  $\square$

Despite the resemblance between equations (3) and (7), the latter is not a subset convolution because of the sign factors. Therefore, we cannot use the algorithm from [10] in this case. We now show to the complexity analysis.

**Lemma 8.** *Given the values of the partial determinants  $\det(A_t, Y_t)$  and  $\det(A_{T_{c_j}}, Y_{c_j})$ , we can evaluate equation (7) for all  $Y$  satisfying (2) in  $\tilde{O}(k(n + 3^\omega))$ .*

*Proof.* We will first express equation (7) in a similar format as formula (5) of Lemma 4. To simplify the notation, let  $\mathcal{Y} = (Y_0, Y_1, \dots, Y_k)$ . For each  $j$  let  $Y_0^j = Y_0 \cup Y_1 \cup \dots \cup Y_j$ , and observe that  $\operatorname{sgn}(\mathcal{Y}) = \prod_j \operatorname{sgn}(Y_0^{j-1}, Y_j)$ . Then equation (7) can be rewritten as:

$$D(\bar{Y}) = \operatorname{sgn}(\mathcal{D}) \sum_{\bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_k = \bar{Y}} \prod_{j=0}^k S_j(\bar{Y}_0^{j-1}, \bar{Y}_j) D_j(\bar{Y}_j)$$

where  $\bar{Y}_0, \dots, \bar{Y}_k \subseteq \chi(t)$  and

$$\Delta := \chi(T_t) \setminus \chi(t) \quad \Delta_0 := \emptyset \quad \Delta_j := \chi(T_{c_j}) \setminus \chi(t)$$

$$D(\bar{Y}) := \det(A_{T_t}, \bar{Y} \cup \Delta), \quad D_0(\bar{Y}_0) := \det(A_t, \bar{Y}_0 \cup \Delta_0), \quad D_j(\bar{Y}_j) := \det(A_{T_{c_j}}, \bar{Y}_j \cup \Delta_j)$$

$$\bar{Y}_0^j = \bar{Y}_0 \cup \dots \cup \bar{Y}_j \quad \Delta_0^j := \Delta_0 \cup \dots \cup \Delta_j$$

$$S_j(\bar{Y}_0^{j-1}, \bar{Y}_j) := \operatorname{sgn}(\bar{Y}_0^{j-1} \cup \Delta_0^{j-1}, \bar{Y}_j \cup \Delta_j)$$

For each  $0 \leq l \leq k$ , and for each  $\bar{Y}_0^l \subseteq \chi(t)$ , let

$$D_0^l(\bar{Y}_0^l) = \text{sgn}(\mathcal{D}) \sum_{\bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_l = \bar{Y}_0^l} \prod_{j=0}^l S_j(\bar{Y}_0^{j-1}, \bar{Y}_j) D_j(\bar{Y}_j)$$

Note that  $D_0^k(\bar{Y}) = D(\bar{Y})$ , and thus it is enough to compute  $D_0^l$  for all  $l$ . We can do this recursively, observing that  $D_0^0(\bar{Y}) = \text{sgn}(\mathcal{D}) D_0(\bar{Y})$  and

$$(8) \quad D_0^{l+1}(\bar{Y}_0^{l+1}) = \sum_{\bar{Y}_0^l \sqcup \bar{Y}_{l+1} = \bar{Y}_0^{l+1}} S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1}) D_0^l(\bar{Y}_0^l) D_{l+1}(\bar{Y}_{l+1})$$

We reduced the problem to evaluating the above formula, and we will show that for each  $l$  we can do this in  $\tilde{O}(n + 3^\omega)$ . Assume for now that the signs  $S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1})$  are known. Then for each  $\bar{Y}_0^{l+1}$  of cardinality  $i$ , we can evaluate (8) in  $O(2^i)$ . Thus, for all  $\bar{Y}_0^{l+1}$  we require  $O(\sum_i \binom{w}{i} 2^i) = O(3^w)$ , where  $w = |\chi(t)|$ . We will see that after a precomputation that takes  $\tilde{O}(n)$ , we can obtain  $S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1})$  in  $\tilde{O}(1)$ , which will complete the proof.

Observe that

$$S_{l+1}(\bar{Y}_0^l, \bar{Y}_{l+1}) = \text{sgn}(\bar{Y}_0^l, \bar{Y}_{l+1}) \text{sgn}(\bar{Y}_0^l, \Delta_{l+1}) \text{sgn}(\Delta_0^l, \bar{Y}_{l+1}) \text{sgn}(\Delta_0^l, \Delta_{l+1}).$$

Note that the last factor does not depend on the partition and it can be precomputed in  $O(n)$ . Also note that the first factor can be computed in  $O(\omega) = \tilde{O}(1)$ , so we can ignore it. We are left with the second and third factor.

For each  $x \in \chi(t)$ , let

$$N_x^{\Delta_{l+1}} = |\{y \in \Delta_{l+1} : x > y\}|.$$

We can precompute  $N_x^{\Delta_{l+1}}$  for all  $x$  in  $O(\omega n) = \tilde{O}(n)$ . Note that  $\text{sgn}(\bar{Y}_0^l, \Delta_{l+1}) = (-1)^N$  where  $N = \sum_{x \in \bar{Y}_0^l} N_x^{\Delta_{l+1}}$ . Thus, after the precomputation, we can obtain this factor in  $O(\omega) = \tilde{O}(1)$ . A similar procedure can be done for  $\text{sgn}(\Delta_0^l, \bar{Y}_{l+1})$ .  $\square$

**Theorem 9.** *Let  $M$  be a matrix with associated column graph  $G^X$ . Let  $(T, \chi)$  be a tree decomposition of  $G^X$  of width  $\omega$ . Then we can compute  $\text{Det}(M)$  in  $\tilde{O}(n^2 + n 4^\omega)$ .*

*Proof.* There are two changes with respect to the proof of Theorem 5. First, in the base case we need to compute the determinant of all submatrices of  $M_0$ . Using an expansion by minors as in the proof of Lemma 2, we can do this in  $\tilde{O}(4^\omega)$ , i.e., the same as for the permanent. Second, for the recursion formula we use Lemma 8. This increases the time per node from  $\tilde{O}(k_t 2^\omega)$  to  $\tilde{O}(k_t(n + 3^\omega))$ . Therefore, the overall cost is  $\tilde{O}(n^2 + n 4^\omega)$ .  $\square$

We conclude this section by presenting an open question. Given the resemblance in the definition of the permanent and the determinant it is not surprising that they can be computed using very similar tree decomposition methods. The *immanant* of a matrix is another closely related notion:

$$\text{Imm}_\lambda(M) := \sum_{\pi} \chi_\lambda(\pi) \prod_{a \in A} M_{a, \pi(a)}$$

where the sum is over all bijections  $\pi : A \rightarrow X$  and  $\chi_\lambda$  is an irreducible character of the symmetric group. The immanant reduces to the permanent when  $\chi_\lambda$  is the trivial character, and it reduces to the determinant when  $\chi_\lambda$  is the sign character. The computational complexity of immanants has been analyzed in e.g., [13]. A natural question that arises is whether a tree decomposition method can be used to compute them. We remark that the recursion in (7) does not hold for the immanant as  $\chi_\lambda$  is not necessarily multiplicative.

**Question.** *Given a matrix  $M$  of bounded treewidth, can we compute  $\text{Imm}_\lambda(M)$  in polynomial time? In particular, can this be done if either the height or the width of the Young diagram is bounded?*

#### 4. BIPARTITE DECOMPOSITIONS

In the previous section we showed a decomposition method based on the column graph  $G^X$ . We showed that we can compute the permanent in  $\tilde{O}(n 4^{\omega_X})$ , where  $\omega_X$  is the treewidth of  $G^X$ . In this section we will extend this decomposition method to work with the bipartite graph  $G$  (see Definition 2.2). We will show that we can compute the permanent in  $\tilde{O}(n 2^\omega)$ , where  $\omega$  is the treewidth of  $G$ . A Matlab implementation of our algorithm is available in [www.mit.edu/~diegcif](http://www.mit.edu/~diegcif).

Let  $G$  be the bipartite graph of  $M$ . As in the previous sections, we index the rows with a set  $A$  and the columns with  $X$ . We now rephrase the definition of a tree decomposition of  $G$ . A *bipartite decomposition* of  $G$  is a tuple  $(T, \alpha, \chi)$ , where  $T$  is a rooted tree and  $\alpha : T \rightarrow \{0, 1\}^A$ ,  $\chi : T \rightarrow \{0, 1\}^X$  assign some  $\alpha(t) \subseteq A$  and  $\chi(t) \subseteq X$  to each node  $t$  of  $T$ , that satisfies the following conditions.

- i-1. The union of  $\{\alpha(t)\}_{t \in T}$  is the whole row set  $A$ .
- i-2. The union of  $\{\chi(t)\}_{t \in T}$  is the whole column set  $X$ .
- ii. For every edge  $(a, x)$  of  $G$  there exists a node  $t$  of  $T$  with  $a \in \alpha(t)$ ,  $x \in \chi(t)$ .
- iii-1. For every  $a \in A$  the set  $\{t : a \in \alpha(t)\}$  forms a subtree of  $T$ .
- iii-2. For every  $x \in X$  the set  $\{t : x \in \chi(t)\}$  forms a subtree of  $T$ .

The width  $\omega$  of the decomposition is the largest of  $|\alpha(t)| + |\chi(t)|$  among all nodes  $t$ . Note that the above literals are consistent with the ones in Definition 2.1.

As before, we now present an example to illustrate the use of the bipartite graph for computing the permanent.

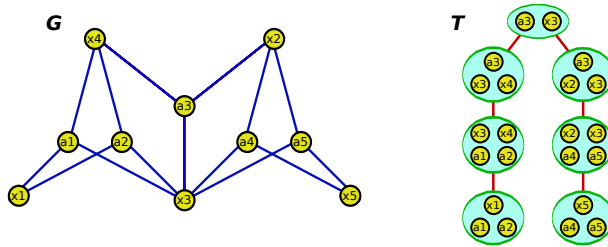


FIGURE 4. Bipartite graph  $G$  of the matrix of Figure 3 and a tree decomposition  $T$ .



**Example 4.1.** Consider again the matrix  $M$  of Figure 3. Note that  $\text{Perm}(M)$  can also be expressed in the following form:

$$\begin{aligned} \text{Perm}(M) &= \text{perm}(\{a_1, a_2\}, \{x_1, x_4\}) \text{perm}(\{a_3, a_4, a_5\}, \{x_2, x_3, x_5\}) \\ &\quad + \text{perm}(\{a_1, a_2, a_3\}, \{x_1, x_3, x_4\}) \text{perm}(\{a_4, a_5\}, \{x_2, x_5\}) \\ &\quad - M_{a_3, x_3} \text{perm}(\{a_1, a_2\}, \{x_1, x_4\}) \text{perm}(\{a_4, a_5\}, \{x_2, x_5\}) \\ \text{perm}(\{a_1, a_2, a_3\}, \{x_1, x_3, x_4\}) &= M_{a_3, x_3} \text{perm}(\{a_1, a_2\}, \{x_1, x_4\}) + M_{a_3, x_4} \text{perm}(\{a_1, a_2\}, \{x_1, x_3\}) \\ \text{perm}(\{a_3, a_4, a_5\}, \{x_2, x_3, x_5\}) &= M_{a_3, x_3} \text{perm}(\{a_4, a_5\}, \{x_2, x_5\}) + M_{a_3, x_2} \text{perm}(\{a_4, a_5\}, \{x_3, x_5\}) \end{aligned}$$

To evaluate the above formula we need to compute four  $2 \times 2$  permanents, and we need in total 16 multiplications. It turns out that this formula arises by considering the tree decomposition of the bipartite graph shown in Figure 4.

**4.1. Bipartite decomposition algorithm.** Algorithm 2 presents our dynamic programming method to compute  $\text{Perm}(M)$  using a bipartite decomposition. As for Algorithm 1, for each node  $t$  we compute a table  $P_t$ , following a topological ordering of the tree. The permanent of  $M$  is in the table corresponding to the root. There are two main routines: SUBPERM computes the permanents of all submatrices, and EVALRECURSION evaluates a recursion formula, which is slightly more complex than the one of Algorithm 1.

As before, the values  $P_t(\bar{D}, \bar{Y})$  computed correspond to a partial permanent of the matrix. Consider the collection

$$S = \{(D, Y) : \alpha(T_t) \setminus \alpha(t) \subseteq D \subseteq \alpha(T_t), \chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t)\}.$$

Observe that  $(D, Y) \in S$  is completely determined by  $(D \cap \alpha(t), Y \cap \chi(t))$ . Therefore, if we let  $\bar{D} = D \cap \alpha(t)$ ,  $\bar{Y} := Y \cap \chi(t)$ , there is a one to one correspondence between  $S$ , and the collection  $\bar{S} := \{(\bar{D}, \bar{Y}) : \bar{D} \subseteq \alpha(t), \bar{Y} \subseteq \chi(t)\}$ . The partial permanents that we are interested in are

$$P_t(\bar{D}, \bar{Y}) := \text{perm}(D, Y) = \text{perm}(\bar{D} \cup (\alpha(T_t) \setminus \alpha(t)), \bar{Y} \cup (\chi(T_t) \setminus \chi(t))).$$

For the root node  $P_{\text{root}}(\alpha(\text{root}), \chi(\text{root})) = \text{perm}(A, X) = \text{Perm}(M)$ .

**4.2. Recursion formula.** The recursion formula that method EVALRECURSION of Algorithm 2 evaluates is given in the following lemma.

**Lemma 10.** *Let  $M$  be a matrix with associated bipartite graph  $G$ . Let  $(T, \alpha, \chi)$  be a bipartite decomposition of  $G$ . Let  $t$  be an internal node of  $T$ , let  $T_t \subseteq T$  denote the subtree rooted in  $t$ , and let  $D, Y$  be such that*

$$(9) \quad \alpha(T_t) \setminus \alpha(t) \subseteq D \subseteq \alpha(T_t), \quad \chi(T_t) \setminus \chi(t) \subseteq Y \subseteq \chi(T_t), \quad |D| = |Y|$$

*Let  $t_{c_1}, \dots, t_{c_k}$  be the children of  $t$ . Then*

$$(10) \quad \text{perm}(D, Y) = \sum_{\mathcal{D}, \mathcal{Y}} \text{perm}(D_t, Y_t) \prod_{j=1}^k (-1)^{|D_{t_{c_j}}|} \text{perm}(D_{t_{c_j}}, Y_{t_{c_j}}) \text{perm}(D_{c_{c_j}}, Y_{c_{c_j}})$$

**Algorithm 2** Permanent with bipartite decomposition**Input:** Matrix  $M$  and tree decomposition  $(T, \alpha, \chi)$  of bipartite graph  $G(M)$ **Output:** Permanent of  $M$ 


---

```

1: procedure BIPARTPERM( $M, T, \alpha, \chi$ )
2:    $order :=$  topological ordering of  $T$  starting from its leaves
3:   for  $t$  in  $order$  do
4:      $Q_t := \text{SUBPERMS}(t, M)$ 
5:     if  $t$  is a leaf then
6:        $P_t := Q_t$ 
7:     else
8:        $c_1, \dots, c_k :=$  children of  $t$ 
9:        $P_t := \text{EVALRECURSION}(t, Q_t, P_{c_1}, \dots, P_{c_k})$ 
10:  return  $P_{root}(\alpha(root), \chi(root))$ 
11: procedure SUBPERMS( $t, M$ )
12:   $Q_t(\bar{D}, \bar{Y}) := \text{perm}(\bar{D}, \bar{Y})$  for all  $\bar{D} \subseteq \alpha(t), \bar{Y} \subseteq \chi(t)$ 
13: procedure EVALRECURSION( $t, Q_t, P_{c_1}, \dots, P_{c_k}$ )
14:  for  $c_j$  child of  $t$  do
15:     $\Delta_j^\alpha := \alpha(c_j) \setminus \alpha(t), \quad \Lambda_j^\alpha := \alpha(c_j) \cap \alpha(t)$ 
16:     $\Delta_j^\chi := \chi(c_j) \setminus \chi(t), \quad \Lambda_j^\chi := \chi(c_j) \cap \chi(t)$ 
17:     $Q_{tc_j}(\bar{D}, \bar{Y}) := (-1)^{|\bar{D}|} Q_t(\bar{D}, \bar{Y})$  for all  $\bar{D} \subseteq \Lambda_j^\alpha, \bar{Y} \subseteq \Lambda_j^\chi$ 
18:     $Q_{cc_j}(\bar{D}, \bar{Y}) := P_{c_j}(\bar{D} \cup \Delta_j^\alpha, \bar{Y} \cup \Delta_j^\chi)$  for all  $\bar{D} \subseteq \Lambda_j^\alpha, \bar{Y} \subseteq \Lambda_j^\chi$ 
19:   $P_t := \text{SUBSETCONVOLUTION}(Q_t, Q_{tc_1}, Q_{cc_1}, \dots, Q_{tc_k}, Q_{cc_k})$ 
20: procedure SUBSETCONVOLUTION( $P_0, P_1, \dots, P_{2k}$ )
21:   $P(\bar{D}, \bar{Y}) := \sum_{\substack{\bar{D}_0 \sqcup \dots \sqcup \bar{D}_{2k} = \bar{D} \\ \bar{Y}_0 \sqcup \dots \sqcup \bar{Y}_{2k} = \bar{Y}}} P_0(\bar{D}_0, \bar{Y}_0) P_1(\bar{D}_1, \bar{Y}_1) \dots P_{2k}(\bar{D}_{2k}, \bar{Y}_{2k})$ 

```

---

where  $\text{perm}(\cdot, \cdot)$  is as in (1) and the sum is over all  $\mathcal{D} = (D_t, D_{tc_1}, \dots), \mathcal{Y} = (Y_t, Y_{tc_1}, \dots)$  satisfying:

$$\begin{aligned}
D &= D_t \sqcup (D_{tc_1} \sqcup D_{cc_1} \sqcup \dots \sqcup D_{tc_k} \sqcup D_{cc_k}) \\
Y &= Y_t \sqcup (Y_{tc_1} \sqcup Y_{cc_1} \sqcup \dots \sqcup Y_{tc_k} \sqcup Y_{cc_k}) \\
\alpha(T_{c_j}) \setminus \alpha(t) &\subseteq D_{cc_j} \subseteq \alpha(T_{c_j}) & D_t &\subseteq \alpha(t) & D_{tc_j} &\subseteq \alpha(t) \cap \alpha(t_{c_j}) \\
\chi(T_{c_j}) \setminus \chi(t) &\subseteq Y_{cc_j} \subseteq \chi(T_{c_j}) & Y_t &\subseteq \chi(t) & Y_{tc_j} &\subseteq \chi(t) \cap \chi(t_{c_j}).
\end{aligned}$$

To prove this lemma we need some additional notation. We view a bijection  $\pi : D \rightarrow Y$  as a subset of  $D \times Y$ , by identifying it with the set  $\{(a, \pi(a)) : a \in D\}$ . For a given node  $t$  and for some  $D, Y$  satisfying (9), we denote

$$\text{perm}^*(D, Y) := \sum_{\pi} \prod_{a \in D} M_{a, \pi(a)}$$

where the sum is over all bijections  $\pi : D \rightarrow Y$  such that

$$(11) \quad \pi \cap (\alpha(t) \times \chi(t)) = \emptyset.$$

We now show a different recursion formula, which is closer to the one in Lemma 3.

**Lemma 11.** *Following the same notation as above, the following equation holds:*

$$(12) \quad \text{perm}(D, Y) = \sum_{Y_t, Y_{c_j}, D_t, D_{c_j}} \text{perm}(D_t, Y_t) \prod_{j=1}^k \text{perm}^*(D_{c_j}, Y_{c_j})$$

where the sum is over all  $Y_t, Y_{c_j}, D_t, D_{c_j}$  such that

$$(13a) \quad D = D_t \sqcup (D_{c_1} \sqcup \dots \sqcup D_{c_k})$$

$$(13b) \quad Y = Y_t \sqcup (Y_{c_1} \sqcup \dots \sqcup Y_{c_k})$$

$$(13c) \quad \alpha(T_{c_j}) \setminus \alpha(t) \subseteq D_{c_j} \subseteq \alpha(T_{c_j}) \quad D_t \subseteq \alpha(t)$$

$$(13d) \quad \chi(T_{c_j}) \setminus \chi(t) \subseteq Y_{c_j} \subseteq \chi(T_{c_j}) \quad Y_t \subseteq \chi(t).$$

*Proof.* Let  $\pi : D \rightarrow Y$  be a matching, which we view as a subset of  $D \times Y$ . Note that

$$\begin{aligned} D &= (D \cap \alpha(t)) \sqcup (D \cap \alpha(T_{c_1}) \setminus \alpha(t)) \sqcup \dots \sqcup (D \cap \alpha(T_{c_k}) \setminus \alpha(t)) \\ Y &= (Y \cap \chi(t)) \sqcup (Y \cap \chi(T_{c_1}) \setminus \chi(t)) \sqcup \dots \sqcup (Y \cap \chi(T_{c_k}) \setminus \chi(t)) \end{aligned}$$

Let's decompose  $\pi$  in a similar way as above. Let  $\pi_t$  be the submatching of  $\pi$  with domain contained in  $D \cap \alpha(t)$  and range contained in  $Y \cap \chi(t)$ . Equivalently,  $\pi_t = \pi \cap (D \cap \alpha(t)) \times (Y \cap \chi(t))$ . Observe that if some  $a \in D \cap \alpha(t)$  is not in the domain of  $\pi_t$ , then  $a \in \alpha(T_c), \pi(a) \in \chi(T_c)$  for some child  $c$ . The reason is that by definition of tree decomposition, there must be some node  $t_p$  with  $\pi(a) \in \chi(t_p), a \in \alpha(t_p)$ . However, the assumption on  $a$  says that  $\pi(a) \in \chi(T_t) \setminus \chi(t)$  and thus we must have  $t_p \in T_c$  for some  $c$ . Similarly, if some  $x \in Y \cap \chi(t)$  is not in the range of  $\pi_t$ , then  $x \in \chi(T_c), \pi^{-1}(x) \in \alpha(T_c)$  for some child  $c$ . In the same way, we have the following

$$\begin{aligned} \pi(D \cap \alpha(T_c) \setminus \alpha(t)) &\subseteq D \cap \chi(T_c) \\ \pi^{-1}(Y \cap \chi(T_c) \setminus \chi(t)) &\subseteq Y \cap \alpha(T_c) \end{aligned}$$

The above paragraph implies that we can decompose

$$(14) \quad \pi = \pi_t \sqcup \pi_{c_1} \sqcup \dots \sqcup \pi_{c_k}$$

in such a way that  $\pi_t \subseteq (D \cap \alpha(t)) \times (Y \cap \chi(t))$  and for each child  $c$  we have that  $\pi_c \subseteq (D \cap \alpha(T_c)) \times (Y \cap \chi(T_c))$  and  $\pi_c$  satisfies (11). Moreover, it is easy to see that such decomposition is unique.

Let  $Y_t \subseteq Y \cap \chi(t)$  be the range of  $\pi_t$  and let  $Y_c \subseteq Y \cap \chi(T_c)$  be the range of  $\pi_c$ . Analogously, define  $D_t \subseteq D \cap \alpha(t)$  and  $D_c \subseteq D \cap \alpha(T_c)$  as the domains of  $\pi_t, \pi_c$ . Observe that  $\chi(T_c) \setminus \chi(t) \subseteq Y_c$ , as if  $x \in \chi(T_c) \setminus \chi(t)$  then  $(\pi^{-1}(x), x) \in \pi_c$  by construction, and thus  $x \in Y_c$ . Similarly,  $\chi(T_c) \setminus \chi(t) \subseteq Y_c$ . Therefore, the equations (13) are satisfied.

Thus, for any matching  $\pi \subseteq D \times Y$  there is a unique partition as in (14) such that the ranges and domains  $Y_t, Y_c, D_t, D_c$  satisfy (13). On the other hand, assume that  $Y_t, Y_c, D_t, D_c$  satisfy (13), and we are given some matchings  $\pi_t, \pi_c$  with these ranges and domains and such that (11) holds. Then equations (13) tell us that we can merge

them into a matching  $\pi$  with domain  $D$  and range  $Y$ . Condition (11) ensures we are not overcounting, as it implies that decomposition (14) is unique. These remarks imply equation (12).  $\square$

*Remark 4.1.* Let  $(T^X, \chi)$  be a tree decomposition of the column graph  $G^X$ , and let  $(T, \alpha, \chi)$  be the corresponding decomposition of the bipartite graph  $G$  given in Example 2.3. In such case, the above lemma reduces to Lemma 3.

We now derive the recursion formula in Lemma 10, which follows from the above lemma by using inclusion-exclusion.

*Proof of Lemma 10.* For a child  $c$  of  $t$ , we will show that

$$(15) \quad \text{perm}^*(D_c, Y_c) = \sum_{\substack{D_{tc} \sqcup D_{cc} = D_c \\ Y_{tc} \sqcup Y_{cc} = Y_c \\ D_{tc} \subseteq \alpha(t), Y_{tc} \subseteq \chi(t)}} (-1)^{|D_{tc}|} \text{perm}(D_{tc}, Y_{tc}) \text{perm}(D_{cc}, Y_{cc}).$$

Combining equations (12) and (15) we obtain equation (10), concluding the proof.

Given a matching  $\pi_c : D_c \rightarrow Y_c$ , let  $I(\pi_c) := \pi_c \cap (\alpha(t) \times \chi(t))$  and let  $I_\alpha(\pi_c) \subseteq \alpha(t)$ ,  $I_\chi(\pi_c) \subseteq \chi(t)$  be the domain and range of  $I(\pi_c)$ . For some  $D_{tc} \subseteq D_c \cap \alpha(t)$ ,  $Y_{tc} \subseteq Y_c \cap \chi(t)$  with  $|D_{tc}| = |Y_{tc}|$ , let

$$\text{perm}^*(D_c, Y_c; D_{tc}, Y_{tc}) := \sum_{\substack{\pi_c : D_c \rightarrow Y_c \\ I_\alpha(\pi_c) = D_{tc} \\ I_\chi(\pi_c) = Y_{tc}}} \prod_a M_{a, \pi_c(a)}.$$

Note that  $\text{perm}^*(D_c, Y_c) = \text{perm}^*(D_c, Y_c; \emptyset, \emptyset)$ . Observe now that given matchings  $\pi_{tc} : D_{tc} \rightarrow Y_{tc}$  and  $\pi_{cc} : D_c \setminus D_{tc} \rightarrow Y_c \setminus Y_{tc}$ , we can merge them into a matching  $\pi_c^* : D_c \rightarrow Y_c$  that satisfies  $I_\alpha(\pi_c^*) \supseteq D_{tc}$ ,  $I_\chi(\pi_c^*) \supseteq Y_{tc}$ . Therefore, we have the following equation

$$\text{perm}(D_{tc}, Y_{tc}) \text{perm}(D_c \setminus D_{tc}, Y_c \setminus Y_{tc}) = \sum_{\substack{D_{tc}^* \supseteq D_{tc} \\ Y_{tc}^* \supseteq Y_{tc}}} \text{perm}^*(D_c, Y_c; D_{tc}^*, Y_{tc}^*).$$

Based on the above formula, we can now find  $\text{perm}^*(D_c, Y_c)$  using inclusion-exclusion (or Möbius inversion):

$$\text{perm}^*(D_c, Y_c; \emptyset, \emptyset) = \sum_i \sum_{\substack{D_{tc} \subseteq D_c \cap \alpha(t) \\ Y_{tc} \subseteq Y_c \cap \chi(t) \\ |D_{tc}| = |Y_{tc}| = i}} (-1)^i \text{perm}(D_{tc}, Y_{tc}) \text{perm}(D_c \setminus D_{tc}, Y_c \setminus Y_{tc}).$$

Rewriting the above equation leads to (15), as wanted.  $\square$

**4.3. Complexity analysis.** We just derived the recursion formula (10) which is used in Algorithm 2. As in the proof of Lemma 4, this formula is a subset convolution and thus it can be evaluated efficiently using the algorithm from [10]. The overall running time of Algorithm 2 is stated now.

**Theorem 12.** *Let  $M$  be a matrix with associated bipartite graph  $G$ . Let  $(T, \alpha, \chi)$  be a bipartite decomposition of  $G$  of width  $\omega$ . Then we can compute  $\text{Perm}(M)$  in  $\tilde{O}(n 2^\omega)$ .*

*Proof.* The proof is very similar to the one of Theorem 5. For each node  $t \in T$ , we will compute  $\text{perm}(D, Y)$  for every pair  $D, Y$  that satisfies equation (9). We will show that for each  $t$  we can compute  $\text{perm}(D, Y)$  for all such  $D, Y$  in  $\tilde{O}((k_t + 1)2^\omega)$ , where  $k_t$  is the number of children of  $t$ . Observe that for the root node  $t_r$  we will compute  $\text{Perm}(M) = \text{perm}(\alpha(T_r), \chi(T_r))$ . This will conclude the proof.

The base case is when  $t$  is a leaf of  $T$ , so that  $T_t = \{t\}$ . Let  $M_0$  be the submatrix of  $M$  with rows  $\alpha(t)$  and columns  $\chi(t)$ . We need to obtain the permanent of all submatrices of  $M_0$ . As  $|\alpha(t)| + |\chi(t)| \leq \omega$ , we can do this in  $\tilde{O}(2^\omega)$  using Lemma 2.

Assume now that  $t$  is an internal node of  $T$  with  $k_t$  children and let  $D, Y$  that satisfy (9). Then equation (10) tells us how to find  $\text{perm}(D, Y)$ . Similarly as in Lemma 4, we can evaluate this formula in  $\tilde{O}(k_t 2^\omega)$ , assuming we know the values of the terms in the recursion. Note that we already found  $\text{perm}(D_{cc}, Y_{cc})$  for all children in the recursion. We can find  $\text{perm}(D_t, Y_t)$  for all  $D_t, Y_t$  in  $\tilde{O}(2^\omega)$  in the same way as for the base case, and this includes the values  $\text{perm}(D_{tc}, Y_{tc})$ . Then, it takes  $\tilde{O}(2^\omega + k_t 2^\omega) = \tilde{O}((k_t + 1)2^\omega)$  to compute  $\text{perm}(D, Y)$  for all  $D, Y$ .  $\square$

Similarly as in Theorem 9, we can find an analogous algorithm for the determinant.

**Theorem 13.** *Let  $M$  be a matrix with associated bipartite graph  $G$ . Let  $(T, \alpha, \chi)$  be a bipartite decomposition of  $G$  of width  $\omega$ . Then we can compute  $\text{Det}(M)$  in  $\tilde{O}(n^2 + n 3^\omega)$ .*

*Proof.* We just need to follow the steps of Section 3.5. For instance, the recursion is

$$\det(D, Y) = \sum_{\mathcal{D}, \mathcal{Y}} \text{sgn}(\mathcal{D}) \text{sgn}(\mathcal{Y}) \det(D_t, Y_t) \prod_{j=1}^k (-1)^{|D_{tc_j}|} \det(D_{tc_j}, Y_{tc_j}) \det(D_{cc_j}, Y_{cc_j})$$

where  $\mathcal{Y} = (Y_t, Y_{tc}, Y_{cc})$ ,  $\mathcal{D} = (D_t, D_{tc}, D_{cc})$ . The complexity analysis is basically the same as in the proof of Theorem 9. The base case can be done in  $\tilde{O}(2^\omega)$  using an expansion by minors. The recursion can be evaluated in  $\tilde{O}(k_t(n + 3^\omega))$  in a similar way as in Lemma 8.  $\square$

## 5. MIXED DISCRIMINANT AND HIGHER DIMENSIONS

The mixed discriminant of  $n$  matrices is a common generalization of the permanent and the determinant. As such, it is also hard to compute in the general case. We show now that the techniques presented earlier generalize to compute mixed discriminants. Even more, we show that this method extends to compute similar functions in higher dimensional tensors.

**5.1. Mixed discriminant.** Let  $M$  be a list of  $n$  matrices of size  $n \times n$ . Equivalently, we can think of  $M$  as a  $n \times n \times n$  array. We index the first coordinate with a set  $A$ , and the second and third coordinates with sets  $X^1, X^2$ . The mixed discriminant of  $M$  is given by

$$\text{Disc}(M) := \sum_{\pi^1, \pi^2} \text{sgn}(\pi^1) \text{sgn}(\pi^2) \prod_{a \in A} M_{a, \pi^1(a), \pi^2(a)}$$

where the sum is over all bijections  $\pi^1 : A \rightarrow X^1$  and  $\pi^2 : A \rightarrow X^2$ , and  $\text{sgn}$  is the parity function. For  $a \in A$ , let  $M_a$  denote the  $n \times n$  matrix obtained by fixing the first coordinate. Observe that if  $M_a = m$  for some matrix  $m$  and for all  $a \in A$ , then  $\text{Disc}(M) = n! \text{Det}(m)$ . In the case that  $M_a$  is diagonal for all  $a \in A$ , then  $\text{Disc}(M) = \text{Perm}(D)$  where  $D$  is the matrix obtained by concatenating these diagonals. Some of the properties of mixed discriminant are discussed in [4].

In the case of a  $n \times n$  matrix, a bipartite graph was the natural structure to represent its sparsity. Similarly, if we are given a sparse  $n \times n \times n$  array  $M$ , a natural structure is a *tripartite graph*  $G$ , as follows. Let  $G$  be the graph on  $A \cup X^1 \cup X^2$ , where for each nonzero entry  $M_{a,x_1,x_2}$  we put a triangle  $\{a, x_1, x_2\}$ .

We rephrase the definition of a tree decomposition of a tripartite graph  $G$ . A *tripartite decomposition* of  $G$  is a tuple  $(T, \alpha, \chi^1, \chi^2)$ , where  $T$  is a rooted tree,  $\alpha : T \rightarrow \{0, 1\}^A$ ,  $\chi^1 : T \rightarrow \{0, 1\}^{X^1}$  and  $\chi^2 : T \rightarrow \{0, 1\}^{X^2}$ , that satisfies the following conditions.

- i. The union of  $\{\alpha(t)\}_{t \in T}$  (resp.  $\chi^1, \chi^2$ ) is the whole  $A$  (resp.  $X^1, X^2$ ).
- ii. For every triangle  $(a, x_1, x_2)$  in  $G$  there is a  $t$  with  $(a, x_1, x_2) \in (\alpha \times \chi^1 \times \chi^2)(t)$ .
- iii. For every  $a \in A$  (resp.  $X^1, X^2$ ) the set  $\{t : a \in \alpha(t)\}$  is a subtree of  $T$ .

The width of the decomposition is the largest of  $|\alpha(t)| + |\chi^1(t)| + |\chi^2(t)|$  among all nodes  $t$ . Note that the above literals are consistent with the ones in Definition 2.1. In particular, observe that the second condition does not impose additional constraints due to Lemma 1.

We proceed to extend the previous results to the mixed discriminant. For some sets  $D \subseteq A$ ,  $Y^1 \subseteq X^1$  and  $Y^2 \subseteq X^2$  we denote

$$(16) \quad \text{disc}(D, Y^1, Y^2) := \sum_{\pi^1, \pi^2} \text{sgn}(\pi^1) \text{sgn}(\pi^2) \prod_{a \in D} M_{\pi^1(a) \pi^2(a)}$$

where the sum is over all bijections  $\pi^1 : D \rightarrow Y^1$  and  $\pi^2 : D \rightarrow Y^2$ . This only makes sense if  $|D| = |Y^1| = |Y^2|$ , and otherwise we can define  $\text{disc}(D, Y^1, Y^2) = 0$ .

As for the case of the permanent, the dynamic program to compute  $\text{Disc}(M)$  has two main steps: computing the mixed discriminant of all subarrays of  $M$ , and evaluating some recursion formula. For the first step, it is easy to see that the approach from Lemma 2 extends, as we show now.

**Lemma 14.** *Let  $M_0$  be a  $n_1 \times n_2 \times n_3$  array. Let  $A_0, X_0^1, X_0^2$  be its set of coordinates, and let  $S = \{(D, Y^1, Y^2) \subseteq A_0 \times X_0^1 \times X_0^2 : |D| = |Y^1| = |Y^2|\}$ . We can compute  $\text{disc}(D, Y^1, Y^2)$  for all triples in  $S$  in  $O(n_{\max}^3 2^{n_1+n_2+n_3})$ , where  $n_{\max} = \max\{n_1, n_2, n_3\}$ .*

*Proof.* For  $i = 1, 2, \dots, \min\{n_1, n_2, n_3\}$  let

$$S_i := \{(D, Y^1, Y^2) \subseteq A_0 \times X_0^1 \times X_0^2 : |D| = |Y^1| = |Y^2| = i\}.$$

We can find  $\text{disc}(D, Y^1, Y^2)$  for all  $(D, Y^1, Y^2) \in S_i$  using the values of  $S_{i-1}$  as follows. Let  $a_0$  be the first element in  $D$ , it is easy to see that

$$\text{disc}(D, Y^1, Y^2) = \sum_{x_1 \in Y^1, x_2 \in Y^2} \epsilon(x_1, x_2) M_{a_0, x_1, x_2} \text{disc}(D \setminus a_0, Y^1 \setminus x_1, Y^2 \setminus x_2)$$

where  $\epsilon(x_1, x_2)$  is either  $+1$  or  $-1$ . To be concrete, if we identify  $Y^1, Y^2$  with the set  $\{1, \dots, i\}$ , then  $\epsilon(j_1, j_2) = (-1)^{j_1+j_2}$ . Thus, for each triple  $(D, Y^1, Y^2)$  we just need to loop over  $n_2 n_3$  terms, and for each we need  $O(n_{\max})$  to find  $D \setminus a_0, Y^1 \setminus x_1, Y^2 \setminus x_2$ .  $\square$

The recursion formula we need to evaluate is given in the following lemma.

**Lemma 15.** *Let  $M$  be a list of  $n$  matrices of size  $n \times n$ , with associated tripartite graph  $G$ . Let  $(T, \alpha, \chi^1, \chi^2)$  be a tripartite decomposition of  $G$ . Let  $t$  be an internal node of  $T$ , let  $T_t \subseteq T$  denote the subtree rooted in  $t$ , and let  $D, Y^1, Y^2$  be such that*

$$(17a) \quad \alpha(T_t) \setminus \alpha(t) \subseteq D \subseteq \alpha(T_t), \quad |D| = |Y^1| = |Y^2|$$

$$(17b) \quad \chi^1(T_t) \setminus \chi^1(t) \subseteq Y^1 \subseteq \chi^1(T_t), \quad \chi^2(T_t) \setminus \chi^2(t) \subseteq Y^2 \subseteq \chi^2(T_t)$$

Let  $c_1, \dots, c_k$  be the children of  $t$ . Then

$$(18) \quad \begin{aligned} \text{disc}(D, Y^1, Y^2) &= \sum_{\mathcal{D}, \mathcal{Y}^1, \mathcal{Y}^2} \text{sgn}(\mathcal{Y}^1) \text{sgn}(\mathcal{Y}^2) \text{disc}(D_t, Y_t^1, Y_t^2) \\ &\quad \prod_{j=1}^k (-1)^{|D_{tc_j}|} \text{disc}(D_{tc_j}, Y_{tc_j}^1, Y_{tc_j}^2) \text{disc}(D_{cc_j}, Y_{cc_j}^1, Y_{cc_j}^2) \end{aligned}$$

where  $\text{disc}(\cdot, \cdot, \cdot)$  is as in (16),  $\text{sgn}(\cdot, \cdot)$  as in Definition 3.1, and the sum is over all  $\mathcal{D} = (D_t, D_{tc_1}, \dots)$ ,  $\mathcal{Y}^1 = (Y_t^1, Y_{tc_1}^1, \dots)$ ,  $\mathcal{Y}^2 = (Y_t^2, Y_{tc_1}^2, \dots)$  satisfying:

$$\begin{aligned} Z &= Z_t \sqcup (Z_{tc_1} \sqcup Z_{cc_1} \sqcup \dots \sqcup Z_{tc_k} \sqcup Z_{cc_k}) & \text{where } Z \in \{D, Y^1, Y^2\} \\ \zeta(T_{c_j}) \setminus \zeta(t) &\subseteq Z_{cc_j} \subseteq \zeta(T_{c_j}) \quad Z_t \subseteq \zeta(t) \quad Z_{tc_j} \subseteq \zeta(t) \cap \zeta(T_{c_j}) & \text{where } \zeta \in \{\alpha, \chi^1, \chi^2\}. \end{aligned}$$

*Proof.* Let  $\pi^1 : D \rightarrow Y^1$  and  $\pi^2 : D \rightarrow Y^2$  be matchings. Observe that Lemma 6 says that  $\text{sgn}(\pi^1)$  will factor in the tree decomposition, leading to the term  $\text{sgn}(\mathcal{D})\text{sgn}(\mathcal{Y}^1)$ . Similarly,  $\text{sgn}(\pi^2)$  leads to the term  $\text{sgn}(\mathcal{D})\text{sgn}(\mathcal{Y}^2)$  (note that  $\text{sgn}(\mathcal{D})$  cancels). Therefore, for the rest of the proof we can ignore all sign factors. We can think of the pair  $(\pi^1, \pi^2)$  as a subset of  $D \times Y^1 \times Y^2$ . In a similar way as we did in the proof of Lemma 11, there is a unique decomposition of  $(\pi^1, \pi^2)$  of the form

$$(19) \quad (\pi^1, \pi^2) = (\pi_t^1, \pi_t^2) \sqcup (\pi_{c_1}^1, \pi_{c_1}^2) \sqcup \dots \sqcup (\pi_{c_k}^1, \pi_{c_k}^2)$$

where  $(\pi_t^1, \pi_t^2) \subseteq (\alpha \times \chi^1 \times \chi^2)(t)$ , and for each child  $c$  we have that  $(\pi_c^1, \pi_c^2) \subseteq (\alpha \times \chi^1 \times \chi^2)(T_c)$  and

$$(20) \quad (\pi_c^1, \pi_c^2) \cap (\alpha \times \chi^1 \times \chi^2)(t) = \emptyset.$$

Note that by construction  $\pi_t^1, \pi_t^2$  have the same domain. Let  $D_t$  denote this domain, and let  $Y_t^1, Y_t^2$  denote the respective ranges. Similarly, let  $D_c, Y_c^1, Y_c^2$  be the domain and ranges of  $\pi_c^1, \pi_c^2$ . Then we have

$$(21a) \quad Z = Z_t \sqcup (Z_{c_1} \sqcup \dots \sqcup Z_{c_k}) \quad \text{where } Z \in \{D, Y^1, Y^2\}$$

$$(21b) \quad \zeta(T_c) \setminus \zeta(t) \subseteq Z_c \subseteq \zeta(T_c) \quad Z_t \subseteq \zeta(t) \quad \text{where } \zeta \in \{\alpha, \chi^1, \chi^2\}$$

Thus, for matchings  $(\pi^1, \pi^2) \in D \times Y^1 \times Y^2$  there is a unique partition as in (19) and the corresponding domains and ranges satisfy (21). On the other hand, assume that  $D_t, D_c, Y_t^1, Y_c^1, Y_t^2, Y_c^2$  satisfy (13), and we are given some matchings  $\pi_t^1, \pi_c^1, \pi_t^2, \pi_c^2$  with

these domains and ranges and such that  $(\pi_c^1, \pi_c^2)$  satisfy (20). Then equations (21) tell us that we can merge them into matchings  $\pi^1, \pi^2$  with domain  $D$  and ranges  $Y^1, Y^2$ . Condition (20) ensures we are not overcounting. Then we have

$$(22) \quad \text{disc}(D, Y^1, Y^2) = \sum_{D_t, D_c, Y_t^1, Y_c^1, Y_t^2, Y_c^2} \text{disc}(D_t, Y_t^1, Y_t^2) \prod_{j=1}^k \text{disc}^*(D_{c_j}, Y_{c_j}^1, Y_{c_j}^2)$$

where the sum is over all triples as in (21), and where  $\text{disc}^*(D_c, Y_c^1, Y_c^2)$  is similar to  $\text{disc}(D_c, Y_c^1, Y_c^2)$ , except that it only uses matchings  $(\pi_c^1, \pi_c^2)$  satisfying (20).

Finally, we can obtain  $\text{disc}^*(D_c, Y_c^1, Y_c^2)$  using inclusion-exclusion in a similar way as in the proof of Lemma 10:

$$(23) \quad \text{disc}^*(D_c, Y_c^1, Y_c^2) = \sum_{\substack{D_{tc} \sqcup D_{cc} = D_c, \quad D_{tc} \subseteq \alpha(t) \\ Y_{tc}^1 \sqcup Y_{cc}^1 = Y_c^1, \quad Y_{tc}^1 \subseteq \chi^1(t) \\ Y_{tc}^2 \sqcup Y_{cc}^2 = Y_c^2, \quad Y_{tc}^2 \subseteq \chi^2(t)}} (-1)^{|D_{tc}|} \text{disc}(D_{tc}, Y_{tc}^1, Y_{tc}^2) \text{disc}(D_{cc}, Y_{cc}^1, Y_{cc}^2).$$

Combining equations (22) and (23), we obtain equation (18).  $\square$

We proceed to the complexity analysis.

**Theorem 16.** *Let  $M$  be a list of matrices with associated tripartite graph  $G$ . Let  $(T, \alpha, \chi^1, \chi^2)$  be a tripartite decomposition of  $G$  of width  $\omega$ . Then we can compute  $\text{Disc}(M)$  in  $\tilde{O}(n^2 + n3^\omega)$ .*

*Proof.* The proof is very similar to the one of Theorem 12. For each node  $t \in T$ , we compute  $\text{disc}(D, Y^1, Y^2)$  for every triple  $D, Y^1, Y^2$  that satisfies equation (17). We will show that for each  $t$  we can get  $\text{disc}(D, Y^1, Y^2)$  for all such  $D, Y^1, Y^2$  in  $\tilde{O}((k_t + 1)(n + 3^\omega))$ , where  $k_t$  is the number of children of  $t$ .

The base case is when  $t$  is a leaf of  $T$ . Let  $M_0$  be the subarray of  $M$  given by indices  $(\alpha(t), \chi^1(t), \chi^2(t))$ . Then we need to find the mixed discriminant of all subarrays of  $M_0$ . We can do this in  $\tilde{O}(2^\omega)$  using of Lemma 14.

Assume now that  $t$  is an internal node of  $T$  with  $k_t$  children and let  $D, Y^1, Y^2$  that satisfy (17). Then equation (18) tells us how to find  $\text{disc}(D, Y^1, Y^2)$ . Similarly as in Lemma 8, we can evaluate this formula in  $\tilde{O}(k_t(n + 3^\omega))$ , assuming we know all terms in the recursion. We already found  $\text{disc}(D_{cc}, Y_{cc}^1, Y_{cc}^2)$  for all children in the recursion, and we can find  $\text{disc}(D_t, Y_t^1, Y_t^2)$  for all  $D_t, Y_t^1, Y_t^2$  in  $\tilde{O}(2^\omega)$  in the same way as for the base case. This leads to a bound of  $\tilde{O}((k_t + 1)(n + 3^\omega))$  to compute  $\text{disc}(D, Y^1, Y^2)$  for all  $D, Y^1, Y^2$ .  $\square$

**5.2. Higher dimensions.** It is easy to see that our methods extend to compute generalizations of the permanent and determinant in higher dimensions. We consider a square  $(d+1)$ -dimensional array (or tensor)  $M$  of length  $n$ , i.e., of size  $n \times \dots \times n$  ( $d+1$  times). Here we assume  $d$  to be constant. Let's index the first coordinate of  $M$  with a set  $A$ , and the following coordinates with sets  $X^1, \dots, X^d$ . Consider a function  $F$  of



the form

$$(24) \quad F(M) = \sum_{\pi_1, \dots, \pi_d} \prod_{a \in A} \epsilon_1(\pi_1) \cdots \epsilon_d(\pi_d) M_{a, \pi_1(a), \dots, \pi_d(a)}$$

where the sum is over all bijections  $\pi_l : A \rightarrow X^l$ , and where  $\epsilon_l(\pi_l)$  is either 1 or  $\text{sgn}(\pi_l)$ .

Let's consider two special cases of equation (24). The simplest case is when  $\epsilon_l(\pi_l) = 1$  for all  $l$ . We refer to such function as the  $(d+1)$ -dimensional permanent and we denote it as  $\text{Perm}(M)$  [18]. Some applications of this permanent are shown in [2, 34].

Consider now the case when  $d+1$  is even and  $\epsilon_l(\pi_l) = \text{sgn}(\pi_l)$  for all  $l$ . This is perhaps the simplest generalization of the determinant, and it is usually referred to as the first Cayley hyperdeterminant [14]. Some applications of the hyperdeterminant are shown in [5, 25]. As opposed to the 2-dimensional case, computing the hyperdeterminant is #P-hard [22].

We now proceed to extend our decomposition methods to this setting. We associate a  $(d+1)$ -partite graph  $G$  where for each nonzero entry of  $M$  we put a  $(d+1)$ -clique in the respective coordinates. A tree decomposition of  $G$  can be seen as a tuple  $(T, \alpha, \chi^1, \dots, \chi^d)$ . The width  $\omega$  of the decomposition is the largest of  $|\alpha(t)| + |\chi^1(t)| + \dots + |\chi^d(t)|$  among all nodes  $t$ .

As before, for some sets  $D \subseteq A, Y^l \subseteq X^l$ , we consider the function

$$(25) \quad f(D, Y^1, \dots, Y^d) := \sum_{\pi_1, \dots, \pi_d} \prod_{a \in A} \epsilon_1(\pi_1) \cdots \epsilon_d(\pi_d) M_{a, \pi_1(a), \dots, \pi_d(a)}$$

where the sum is over all bijections  $\pi_l : D \rightarrow Y^l$ .

There are two steps in order to generalize our results to this setting: evaluate  $f$  in all subarrays, and evaluate the recursion formula. For the former, the approach from Lemma 2 (and Lemma 14) has a simple generalization. Indeed, Barvinok shows this for the case of the hyperdeterminant [5]. The proof is the same for an arbitrary function  $F$  as in (24). Thus, we have the following.

**Proposition 17** ([5]). *Let  $M_0$  be a  $(d+1)$ -dimensional array of size  $n_0 \times \dots \times n_d$ , and let  $f$  be as in (25). Let  $A_0, X_0^1, \dots, X_0^d$  be its set of coordinates, and let*

$$S := \{(D, Y^1, \dots, Y^d) \subseteq A_0 \times X_0^1 \times \dots \times X_0^d : |D| = |Y^1| = \dots = |Y^d|\}$$

*We can compute  $f(D, Y^1, \dots, Y^d)$  for all tuples in  $S$  in  $O(n_{\max}^{d+1} 2^{n_0 + \dots + n_d})$ , where  $n_{\max} = \max\{n_0, \dots, n_d\}$ .*

Repeating the same analysis as in the proof of Lemma 15, the recursion formula is:

$$(26) \quad f(D, Y^1, \dots, Y^d) = \sum_{\mathcal{D}, \mathcal{Y}^1, \dots, \mathcal{Y}^d} \delta_1(\mathcal{D}, \mathcal{Y}^1) \cdots \delta_d(\mathcal{D}, \mathcal{Y}^d) f(D_t, Y_t^1, \dots, Y_t^d) \prod_{j=1}^k (-1)^{|D_{tc_j}|} f(D_{tc_j}, Y_{tc_j}^1, \dots, Y_{tc_j}^d) f(D_{cc_j}, Y_{cc_j}^1, \dots, Y_{cc_j}^d)$$

where  $\delta_t(\mathcal{D}, \mathcal{Y}^l)$  is either 1 or  $\text{sgn}(\mathcal{D})\text{sgn}(\mathcal{Y}^l)$  depending on  $\epsilon_t$ , and the sum is over all tuples  $\mathcal{D}, \mathcal{Y}^1, \dots, \mathcal{Y}^d$  such that

$$Z = Z_t \sqcup (Z_{tc_1} \sqcup Z_{cc_1} \sqcup \dots \sqcup Z_{tc_k} \sqcup Z_{cc_k}) \quad \text{where } Z \in \{D, Y^1, \dots, Y^d\}$$

$$\zeta(T_{c_j}) \setminus \zeta(t) \subseteq Z_{cc_j} \subseteq \zeta(T_{c_j}) \quad Z_t \subseteq \zeta(t) \quad Z_{tc_j} \subseteq \zeta(t) \cap \zeta(t_{c_j}) \quad \text{where } \zeta \in \{\alpha, \chi^1, \dots, \chi^d\}$$

The complexity of the decomposition algorithm is as follows.

**Theorem 18.** *Let  $M$  be a square  $(d+1)$ -dimensional array of length  $n$ , with  $(d+1)$ -partite graph  $G$ . Let  $F$  be a generalized determinant/permanent as in (24). Let  $(T, \alpha, \chi^1, \dots, \chi^d)$  be a tree decomposition of  $G$  of width  $\omega$ . Then we can compute  $F(M)$  in  $\tilde{O}(n^2 + n3^\omega)$ .*

*Proof.* The proof is very similar to past ones. For each node  $t$ , we compute  $f(D, Y^1, \dots, Y^d)$  for all valid tuples. We show that for each  $t$  we can do this in  $\tilde{O}((k_t + 1)(n + 3^\omega))$ , where  $k_t$  is the number of children of  $t$ .

The base case, i.e., leaf nodes, reduces to Proposition 17, leading to a bound of  $\tilde{O}(2^\omega)$ .

For an internal node  $t$ , equation (26) tells us how to find  $f(D, Y^1, \dots, Y^d)$ . Similarly as in Lemma 8, we can evaluate this formula in  $O(k_t(n + 3^\omega))$ .  $\square$

For the special case of the permanent, we can give a better bound.

**Theorem 19.** *Let  $M$  be a square  $(d+1)$ -dimensional array of length  $n$ , with  $(d+1)$ -partite graph  $G$ . Let  $(T, \alpha, \chi^1, \dots, \chi^d)$  be a tree decomposition of  $G$  of width  $\omega$ . Then we can compute  $\text{Perm}(M)$  in  $\tilde{O}(n2^\omega)$ .*

*Proof.* If there are no sign factors we can follow the procedure of Lemma 4 for the recursion, leading to a bound of  $\tilde{O}((k_t + 1)2^\omega)$  per node.  $\square$

## 6. MIXED VOLUME OF ZONOTOPES

The mixed volume  $\text{MVol}$  of  $n$  convex bodies  $K_1, \dots, K_n$  in  $\mathbb{R}^n$  is the unique real function that satisfies the following properties.

- $\text{MVol}$  is multilinear and symmetric in its arguments.
- $\text{MVol}(K, \dots, K) = n! \text{vol}(K)$ , where  $\text{vol}$  denotes the volume.

Alternatively, it can be shown that the function  $f(\lambda) := \text{vol}(\sum_i \lambda_i K_i)$  for  $\lambda_i \geq 0$ , is a homogeneous polynomial, and  $\text{MVol}$  is the coefficient of  $\lambda_1 \cdots \lambda_n$ . For more information about mixed volumes, see e.g., [30]. We focus here in the case that all bodies  $K_i$  are zonotopes, which are a special class of polytopes.

### 6.1. Mixed volumes and permanents.

**Definition 6.1.** A *zonotope*  $z$  is a polytope that is a Minkowski sum of line segments, i.e., it has the form

$$z = [0, 1]z_1 + [0, 1]z_2 + \dots + [0, 1]z_m = \{r_1 z_1 + \dots + r_m z_m : 0 \leq r_i \leq 1\}$$

where  $z_i \in \mathbb{R}^n$  are vectors. In case  $z_1, \dots, z_m$  are linearly independent, we say that  $z$  is a *parallelotope*.

The mixed volume of zonotopes has a simple description as follows.

**Proposition 20.** Let  $z^i = \sum_{j \in J_i} [0, 1]z_j^i$  be a zonotope, for  $i = 1, \dots, n$ . Then

$$(27) \quad \text{MVol}(z^1, \dots, z^n) = \sum_{j_1 \in J_1, \dots, j_n \in J_n} |\text{Det}(z_{j_1}^1, z_{j_2}^2, \dots, z_{j_n}^n)|$$

*Proof.* The multilinearity of the mixed volume implies

$$\text{MVol}(z^1, \dots, z^n) = \sum_{j_1 \in J_1, \dots, j_n \in J_n} \text{MVol}([0, 1]z_{j_1}^1, \dots, [0, 1]z_{j_n}^n).$$

Thus, we just need to argue that

$$\text{MVol}([0, 1]z_{j_1}^1, \dots, [0, 1]z_{j_n}^n) = |\text{Det}(z_{j_1}^1, \dots, z_{j_n}^n)|$$

which follows by noting that  $\sum_i [0, 1]\lambda_i z_{j_i}^i$  is a parallelepiped with sides  $\lambda_i z_{j_i}^i$ , and thus its volume is given by the absolute volume of the determinant.  $\square$

The mixed volume of  $n$  parallelotopes reduces to a permanent when their main axes are aligned, as shown now.

**Corollary 21.** Let  $u_1, \dots, u_n \in \mathbb{R}^n$  and let  $M \in \mathbb{R}_{\geq 0}^{n \times n}$  be a nonnegative matrix. Let  $z^i = \sum_j [0, 1]M_{i,j}u_j$  be a zonotope. Then

$$\text{MVol}(z^1, \dots, z^n) = |\text{Det}(u_1, \dots, u_n)| \text{Perm}(M).$$

*Proof.* We just need to use equation (27), and cancel out all terms that contain a repeated vector  $u_j$ , as the determinant is zero. The remaining terms have  $|\text{Det}(u_1, \dots, u_n)|$  as a factor and we get the desired formula.  $\square$

**6.2. Graph representation.** To use a decomposition method for mixed volumes we need to have a graph description of the zonotopes. We consider now two different graphs that can be associated to a set of zonotopes, and more generally to polytopes. The first one is a bipartite graph that can be thought of as the analogue of the bipartite graph of a matrix. The second one has to do with the sparsity in the standard basis representation and it is a more intuitive notion for general polytopes.

**Definition 6.2.** Let  $Q$  be a set of  $n$  polytopes in  $\mathbb{R}^n$ . Let  $U$  denote the set of all vectors (up to scaling) that are parallel to some edge in  $Q$ . We refer to  $U$  as the *edge directions* of  $Q$ . The *edge graph*  $G(Q)$  is a bipartite graph with vertices  $Q \cup U$  and edges  $(q, u)$  if  $q$  contains an edge parallel to  $u$ .

**Definition 6.3.** Let  $Q$  be a set of  $n$  polytopes in  $\mathbb{R}^n$ . Let  $X = \{x_1, \dots, x_n\}$  denote the coordinates. The *coordinates graph*  $G^X(Q)$  has  $X$  as vertex set, and for each polytope  $q \in Q$  we form a clique in all its non constant components. Note that if  $q = \sum_j [0, 1]z_j$  is a zonotope, we form a clique in the nonzero coordinates of  $\sum_j z_j$ .

*Remark 6.1.* Note that the edge graph  $G(Q)$  is invariant under affine transformations of  $Q$ , whereas the coordinates graph  $G^X(Q)$  is not.

**Example 6.1** (Zonotopes of bounded treewidth). Consider the following zonotopes:

$$(28a) \quad z^1 = [0, 1](a_1 e_n + e_1) + [0, 1]e_1$$

$$(28b) \quad z^i = [0, 1](a_i e_n + e_i - e_{i-1}) + [0, 1](e_i - e_{i-1}) \quad \text{for } i = 2, \dots, n-1$$

$$(28c) \quad z^n = [0, 1](a_n e_n - e_{n-1})$$

where  $\{e_i\}_i$  is the canonical basis and  $a_i \in \mathbb{Z}$  are some integers.

Note that the segments of the above zonotopes are all nonparallel (there are  $2n + 1$  edge directions). This means that the edge graph  $G$  has  $n$  connected components, one for each of the zonotopes, and each of component is either a 2-path or a 1-path. Thus,  $G$  has treewidth 1. As for the coordinates graph  $G^X$ , it is the union of the triangles:  $X_i := \{x_i, x_{i+1}, x_n\}$ . It is easy to see that  $G^X$  has treewidth 2.

The following example shows the relationship between these graphs and the matrix graphs we used before.

**Example 6.2** (Relationship with matrix graphs). Let  $z^i = \sum_j [0, 1]M_{i,j}u_j$  be zonotopes as in Corollary 21. The edge graph  $G$  of the zonotopes has vertices  $Z \cup U$  where  $Z = \{z^i\}_i$  and  $U = \{u_j\}_j$ , and it has an edge  $(z^i, u_j)$  whenever  $M_{i,j} \neq 0$ . If we replace  $Z$  with the row set and  $U$  with the column set, this is precisely the bipartite graph of matrix  $M$ .

On the other hand, the coordinates graph  $G^X$  depends on the sparsity structure of vectors  $U$ . Assume now that  $U = \{e_j\}_j$  is the canonical basis. Then  $G^X$  has an edge  $(x_j, x_k)$  whenever there is some  $z^i$  with  $M_{i,j} \neq 0$  and  $M_{i,k} \neq 0$ . This corresponds to the column graph of  $M$ .

Because of the above example it is expected that the tree decomposition methods derived for the permanent should allow to compute mixed volumes of certain families of zonotopes. Indeed, we show now if *there are few edge directions* and the edge graph  $G$  has bounded treewidth then the mixed volume can be computed efficiently.

**Theorem 22.** *Let  $Z$  be a set of  $n$  zonotopes in  $\mathbb{R}^n$ . Let  $U$  be the set of edge directions of  $Z$ , and assume that  $d := |U| - n$  is constant. Let  $G$  denote the edge graph of  $Z$ . Given a tree decomposition of  $G$  of width  $\omega$ , we can compute  $\text{MVol}(Z)$  in  $\tilde{O}(n^{d+3} + n^{d+1} 2^\omega)$ .*

*Proof.* If  $|U| < n$  it follows from (27) that  $\text{MVol}(Z) = 0$ . If  $|U| = n$ , then Corollary 21 tells us that we just need to compute the determinant of the  $u_i$ 's and the permanent of some matrix  $M$ . We can find the determinant in  $O(n^3)$  with linear algebra. For the permanent, as the edge graph  $G$  corresponds to the bipartite graph of  $M$ , we can use Theorem 12. Thus, we can find this permanent in  $\tilde{O}(n 2^\omega)$ .

If  $|U| > n$ , let  $W \subseteq U$  of size  $n$ . For each  $z \in Z$  assume  $z = \sum_{u \in U} [0, 1]c_u u$  for some scalars  $c_u$ . Let  $z_W = \sum_{u \in W} [0, 1]c_u u$ , and let  $Z_W = \{z_W : z \in Z\}$ . It follows from equation (27) that

$$\text{MVol}(Z) = \sum_{W \subseteq U, |W|=n} \text{MVol}(Z_W).$$

For each such  $W$  the associated graph is a subgraph of  $G$ , so we can compute  $\text{MVol}(Z_W)$  in  $\tilde{O}(n^3 + n 2^\omega)$ . As there are  $\binom{n+d}{n} = O(n^d)$  possible  $W$ , the result follows.  $\square$

**Example 6.3** (Zonotopes with few edge directions). Consider the following zonotopes:

$$z^i = [0, 1]a_i e_i + [0, 1]b_i e \quad \text{for } i = 1, \dots, n$$

where  $\{e_i\}_i$  is the canonical basis,  $e := \sum_j e_j = (1, \dots, 1)$  and  $a_i, b_i \in \mathbb{Z}$  are some integers. Observe that there are  $n + 1$  edge directions:  $e_1, \dots, e_n, e$ . Also note that the edge graph  $G$  is a tree, consisting of pairs  $(z^i, e_i)$  and  $(z^i, e)$ . Therefore, Theorem 22 says that we can compute the mixed volume of these polytopes in polynomial time.

As an application, recall that Bernstein's Theorem [9] gives a correspondence between mixed volumes and the roots of systems of polynomials. This allows us to conclude that we can efficiently count the number of solutions of the following system of equations:

$$0 = c_{i,1} + c_{i,2} x_i^{a_i} + c_{i,3} \prod_j x_j^{b_i} + c_{i,4} x_i^{a_i} \prod_j x_j^{b_i} \quad \text{for } i = 1, \dots, n.$$

**6.3. Hardness result.** Theorem 22 shows that it is possible to exploit tree decompositions for mixed volume computations. However, it restricts the zonotopes to have a small number of edge directions. This is a strong requirement which is not satisfied in many cases (such as in Example 6.1). Unfortunately, we will see that we need this condition. We remark that the same condition appears in discrete optimization, where it allows to derive (strongly) polynomial time algorithms for certain discrete convex optimization problems [28].

We show now that computing the mixed volume of the zonotopes in Example 6.1 is #P-hard. This shows that mixed volumes of zonotopes continue to be hard, even if both  $G$  and  $G^X$  have bounded treewidth. We use a similar reduction as in [19], where they prove that the volume of zonotopes is #P-hard.

**Lemma 23.** *The determinant of the following  $n \times n$  matrix is  $s_1 + s_2 + \dots + s_n$ .*

$$M = \begin{bmatrix} 1 & -1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -1 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1 & -1 \\ s_1 & s_2 & s_3 & s_4 & \dots & s_{n-1} & s_n \end{bmatrix}$$

*Proof.* If we perform Gaussian elimination we end up with an upper triangular matrix where the diagonal is:  $M_{i,i} = 1$  for  $i < n$  and  $M_{n,n} = s_1 + \dots + s_n$ .  $\square$

**Proposition 24.** *The following problem is #P-hard. Given integers  $a_1, \dots, a_n$ , compute the mixed volume of the  $n$  zonotopes of equations (28).*

*Proof.* We consider the #P-complete problem Subset-Sum: given a set of integers  $A$ , determine the number of subsets  $S \subseteq A$  with sum zero. Let  $k = n - 1$ ,  $A = \{a_1, \dots, a_k\}$  and let  $a_n = \delta$  be a parameter. We will show that the solution to the Subset-Sum problem is given by

$$(29) \quad \frac{1}{2} \text{MVol}(z^1, \dots, z_{\delta=-1}^n) - \text{MVol}(z^1, \dots, z_{\delta=0}^n) + \frac{1}{2} \text{MVol}(z^1, \dots, z_{\delta=1}^n).$$

Let's evaluate equation (27). Consider the following  $n \times (2n - 1)$  matrix.

$$M_\delta = \begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 & -1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 1 & -1 \\ a_1 & 0 & a_2 & 0 & a_3 & 0 & \cdots & a_{k-1} & 0 & a_k & 0 & \delta \end{bmatrix}$$

Observe that columns  $2i - 1$  and  $2i$  of  $M_\delta$  correspond to  $z^i$ , and the last column corresponds to  $z_\delta^n$ . Then formula (27) considers submatrices of  $M_\delta$  that use columns  $j_1, \dots, j_n$  where  $j_i \in \{2i - 1, 2i\}$  for  $i = 1, \dots, k$  and  $j_n = 2n - 1$ . Note now that for any subset  $S \subseteq A$ , there is a natural submatrix  $M_\delta^S$  to consider: if  $a_i \in S$  then  $j_i = 2i - 1$  and otherwise  $j_i = 2i$ . This correspondence is a bijection. Observe also that each submatrix  $M_\delta^S$  has the form of Lemma 23. Thus, we have the following equation:

$$\text{MVol}(z^1, \dots, z_\delta^n) = \sum_{S \subseteq A} \left| \delta + \sum_{a_i \in S} a_i \right|.$$

Finally, observe that for any integer  $s$  we have

$$\frac{1}{2} |(-1) + s| - |s| + \frac{1}{2} |1 + s| = \begin{cases} 1 & \text{if } s = 0 \\ 0 & \text{otherwise} \end{cases}$$

The last two equations imply that (29) indeed counts the subsets  $S$  with sum zero.  $\square$

## REFERENCES

- [1] Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a  $k$ -tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.
- [2] S. V. Avgustinovich. Multidimensional permanents in enumeration problems. *Journal of Applied and Industrial Mathematics*, 4(1):19–20, 2010.
- [3] Nikhil Balaji and Samir Datta. Bounded treewidth and space-efficient linear algebra. In *Theory and Applications of Models of Computation*, volume 9076 of *Lecture Notes in Computer Science*, pages 297–308. Springer International Publishing, 2015.
- [4] R. B.apat. Mixed discriminants of positive semidefinite matrices. *Linear Algebra and its Applications*, 126:107–124, 1989.
- [5] Alexander Barvinok. New algorithms for linear  $k$ -matroid intersection and matroid  $k$ -parity problems. *Mathematical Programming*, 69(1-3):449–470, 1995.
- [6] Alexander Barvinok. Two algorithmic results for the traveling salesman problem. *Mathematics of Operations Research*, 21(1):65–84, 1996.
- [7] Alexander Barvinok. Computing mixed discriminants, mixed volumes, and permanents. *Discrete & Computational Geometry*, 18(2):205–237, 1997.
- [8] Alexander Barvinok and Alex Samorodnitsky. Computing the partition function for perfect matchings in a hypergraph. *Combinatorics, Probability and Computing*, 20(06):815–835, 2011.
- [9] D. N. Bernstein. The number of roots of a system of equations. *Functional Analysis Appl.*, 9(3):1–4, 1975.
- [10] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 67–74. ACM, 2007.
- [11] Jean RS Blair and Barry Peyton. An introduction to chordal graphs and clique trees. In *Graph theory and sparse matrix computation*, pages 1–29. Springer, 1993.

- [12] Hans L Bodlaender and Arie MCA Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.
- [13] Peter Bürgisser. The computational complexity of immanants. *SIAM Journal on Computing*, 30(3):1023–1040, 2000.
- [14] Arthur Cayley. Mémoire sur les hyperdéterminants. *Journal für die reine und angewandte Mathematik*, 30:1–37, 1846.
- [15] Bruno Codenotti, Valentino Crespi, and Giovanni Resta. On the permanent of certain  $(0,1)$  Toeplitz matrices. *Linear Algebra and its Applications*, 267:65–100, 1997.
- [16] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. *Discrete Applied Mathematics*, 108(1):23–52, 2001.
- [17] Rina Dechter. *Constraint processing*. Morgan Kaufmann, 2003.
- [18] Stephen J Dow and Peter M Gibson. Permanents of  $d$ -dimensional matrices. *Linear Algebra and its Applications*, 90:133–145, 1987.
- [19] Martin Dyer, Peter Gritzmann, and Alexander Hufnagel. On the complexity of computing mixed volumes. *SIAM Journal on Computing*, 27(2):356–400, 1998.
- [20] Uffe Flarup, Pascal Koiran, and Laurent Lyaudet. On the expressive power of planar perfect matching and permanents of bounded treewidth matrices. In *Algorithms and Computation*, volume 4835 of *Lecture Notes in Computer Science*, pages 124–136. Springer, 2007.
- [21] Leonid Gurvits. On the complexity of mixed discriminants and related problems. In *Mathematical Foundations of Computer Science*, pages 447–458. Springer, 2005.
- [22] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):1–45, 2013.
- [23] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4):671–697, 2004.
- [24] Pieter W Kasteleyn. Graph theory and crystal physics. *Graph theory and theoretical physics*, 1:43–110, 1967.
- [25] Jean-Gabriel Luque and Jean-Yves Thibon. Hankel hyperdeterminants and Selberg integrals. *Journal of Physics A: Mathematical and General*, 36(19):5267–5292, 2003.
- [26] Klaus Meer. An extended tree-width notion for directed graphs related to the computation of permanents. In *Computer Science Theory and Applications*, pages 247–260. Springer, 2011.
- [27] Henryk Minc. Permanental compounds and permanents of  $(0,1)$ -circulants. *Linear Algebra and its Applications*, 86:11–42, 1987.
- [28] Shmuel Onn and Uriel G Rothblum. Convex combinatorial optimization. *Discrete & Computational Geometry*, 32(4):549–566, 2004.
- [29] Herbert John Ryser. *Combinatorial mathematics*. New York, 1963.
- [30] Rolf Schneider. *Convex bodies: the Brunn-Minkowski theory*, volume 44 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1993.
- [31] Moshe Schwartz. Efficiently computing the permanent and Hafnian of some banded Toeplitz matrices. *Linear Algebra and its Applications*, 430(4):1364–1374, 2009.
- [32] Kristan Temme and Pawel Wocjan. Efficient computation of the permanent of block factorizable matrices. *arXiv preprint arXiv:1208.6589*, 2012.
- [33] HNV Temperley and Michael E Fisher. Dimer problem in statistical mechanics—an exact result. *Philosophical Magazine*, 6(68):1061–1063, 1961.
- [34] Malte C Tichy. Sampling of partially distinguishable bosons and the relation to the multidimensional permanent. *Physical Review A*, 91(2):022316, 2015.
- [35] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.
- [36] Johan MM van Rooij, Hans L Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In *Algorithms-ESA 2009*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009.

- [37] Pascal O Vontobel. The Bethe permanent of a nonnegative matrix. *Information Theory, IEEE Transactions on*, 59(3):1866–1901, 2013.
- [38] Yusuke Watanabe and Michael Chertkov. Belief propagation and loop calculus for the permanent of a non-negative matrix. *Journal of Physics A: Mathematical and Theoretical*, 43(24):242002, 2010.

LABORATORY FOR INFORMATION AND DECISION SYSTEMS (LIDS), MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE MA 02139, USA

*E-mail address:* `diegcif@mit.edu`

LABORATORY FOR INFORMATION AND DECISION SYSTEMS (LIDS), MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE MA 02139, USA

*E-mail address:* `parrilo@mit.edu`